COMPUTERS & SECURITY 70 (2017) 482–499

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

ArOMA: An SDN based autonomic DDoS mitigation framework

Rishikesh Sahay [a,b], Gregory Blanc [a,b], Zonghua Zhang [b,c,*], Hervé Debar [a,b]

[a] Télécom SudParis, Institut Mines-Télécom, France
[b] CNRS UMR 5157 SAMOVAR, France
[c] IMT Lille Douai, Institut Mines-Télécom, France

A R T I C L E I N F O

Article history:
Received 15 November 2016
Received in revised form 7 June 2017
Accepted 9 July 2017
Available online 19 July 2017

Keywords:
DDoS attacks
DDoS mitigation
Software Defined Networking
Anomaly detection
Security policy

A B S T R A C T

Distributed Denial of Service (DDoS) attacks have been the plague of the Internet for more than two decades, despite the tremendous and continuous efforts from both academia and industry to counter them. The lessons learned from the past DDoS mitigation designs indicate that the heavy reliance on additional software modules and dedicated hardware devices seriously impede their widespread deployment. This paper proposes an autonomic DDoS defense framework, called *ArOMA*, that leverages the programmability and centralized manageability features of Software Defined Networking (SDN) paradigm. Specifically, *ArOMA* can systematically bridge the gaps between different security functions, ranging from traffic monitoring to anomaly detection to mitigation, while sparing human operators from non-trivial interventions. It also facilitates the collaborations between ISPs and their customers on DDoS mitigation by logically distributing the essential security functions, allowing the ISP to handle DDoS traffic based on the requests of its customers. Our experimental results demonstrate that, in the face of DDoS flooding attacks, *ArOMA* can effectively maintain the performance of video streams at a satisfactory level.

1. Introduction

Distributed Denial of Service (DDoS) attacks have continuously occurred on the Internet for most of the past three decades, attracting tremendous research efforts from both academia and industry. In particular, flooding-based attacks, such as the ones manipulating UDP, TCP SYN or ICMP packets, are the most prevalent attack variants on the Internet (Arbor Networks). As a matter of fact, DDoS attacks impact not only the victim networks but also the ISP networks, as all the malicious traffic, directed to the victim networks, traverses the ISP network, potentially congesting the links within the ISP network (Eddy et al., 2016). This indicates that the customers and their ISP need to closely collaborate to cope with DDoS attacks. For example, when a customer detects an attack, relevant information about the attack should be shared with the ISP in a timely manner, in order to operate a quick mitigation. Despite a large variety of DDoS mitigation mechanisms, to the best of our knowledge, only a few of them provide a collaborative architecture (Koutepas et al., 2004; Schnakengerg et al., 2001) based on a community of trusted partners. However, this requires all network routers to maintain a partner list, clearly resulting in information exchange overhead. It may

* Corresponding author.
Email addresses: rishikesh.sahay@telecom-sudparis.eu (R. Sahay), gregory.blanc@telecom-sudparis.eu (G. Blanc), zonghua.zhang@imt-lille-douai.fr (Z. Zhang), herve.debar@telecom-sudparis.eu (H. Debar).

http://dx.doi.org/10.1016/j.cose.2017.07.008
0167-4048/© 2017 Elsevier Ltd. All rights reserved.

require, as well, the modification of legacy routers and the deployment of dedicated network devices, incurring necessary, more complex human interventions (Mahimkar et al., 2007).

After a comprehensive survey on the available DDoS mitigation designs, we have identified several requirements to enhance them, and even new designs: (1) easy deployment and operation, avoiding the use of special-purpose software or hardware devices; (2) effective information exchange between the different parties involved (i.e., the ISP and its customers), so that a customer can signal threat information to the ISP at a very early stage of the DDoS attack; (3) timely handling of the customers' requests by the ISP and enforcement of appropriate DDoS mitigation policies; (4) management of the entire lifecycle of the DDoS mitigation scheme in a scalable, adaptive, and automated way, avoiding non-trivial manual effort. Unfortunately, the intrinsic characteristics of today's legacy networking infrastructure make a majority of DDoS mitigation schemes fail to meet the given requirements. The recent development of Software Defined Networking (SDN) motivate us to re-examine such designs (Sahay et al., 2015; Shin et al., 2013), in particular by taking advantage of the clear separation between the network control plane and data plane, as well as, of the programmability of SDN controllers.

To meet all the above-mentioned requirements, we propose *ArOMA*, an SDN based autonomic DDoS mitigation framework. *ArOMA* is intended to systematically and seamlessly integrate different DDoS mitigation modules together, which are distributed across the ISP and its customers, ranging from traffic monitoring to anomaly detection to mitigation. In particular, traffic monitoring and anomaly detection modules are deployed at the customer side, while the mitigation engine runs at the ISP side. Such architecture is motivated by the fact that customers usually have a better opinion on undesired traffic. On the other hand, the ISP has better control over the upstream traffic. To facilitate the collaboration between the ISP and its customers, the intrinsic functions of OpenFlow (OF) switches are explored, such as flow tagging and flow statistics collection. It is worth mentioning that the focus of this paper is on DDoS attacks mitigation rather than their detection. Therefore, we assume that the customer has detected flows suspected of being attacks and shares the alerts with its ISP via a communication channel between the SDN controllers deployed at the ISP and customer sides.

The rest of this paper is organized as follows. Section 2 reviews some related work. A set of key observations and motivations are given in Section 3. An architectural description of *ArOMA*, together with the implementation details of its key functional components, is presented in Section 4. Section 5 develops a use case to illustrate the application of *ArOMA*. The development and implementation of an *ArOMA* prototype using the Ryu SDN controller is reported in Section 6, along with a set of simulation results. In addition, some testbed-based experimental results are reported in Section 7. Section 8 finally concludes the paper.

2. Related work

To date, a large variety of DDoS or DoS mitigation mechanisms have been proposed, covering the entire security life-cycle from prevention and detection to characterization and response. However, few of them have been considered for widespread deployment due to non-trivial deployment overhead and management complexity. We conducted a survey to better understand their shortcomings and classified them into five categories: capability-based, congestion control, traceback, cooperative detection and reaction, and traffic engineering.

2.1. *Capability-based techniques*

These schemes usually require the communicating parties to establish a privileged channel (Liu et al., 2010; Yaar et al., 2004; Yang et al., 2005) beforehand. This means the sender of a message must obtain the permission, e.g., a capability token, from the receiver, in order to send the message. In the presence of DDoS attacks, the privileged traffic is given higher priority than the unprivileged traffic. For example, in the SIFF architecture (Yaar et al., 2004), the sender is required to complete a three-way handshake to get the capability token to initiate the communication. However, it apparently incurs communication latency. Also, part of the capability token is distributed among routers in the network, so the traversed routers need to maintain the state of the flow and part of the capability token, leading to processing overhead and operational complexity. In the *ArOMA* framework, no capability token is required in advance for initiating the communications.

2.2. *Congestion control mechanisms*

A typical scheme is Pushback (Ioannidis and Bellovin, 2002), which instructs a router close to the traffic bottleneck to send messages to the upstream routers, asking to filter the traffic based on a congestion signature and to ensure that the legitimate traffic can flow through the network. To implement Pushback, special devices need to be deployed at each router in the network. The difference between Pushback and *ArOMA* is that *ArOMA* shares the attack information via the communication channel between the SDN controllers instead of the data plane routers. In Hsiao et al. (2013), a trust-based DDoS mitigation framework, called STRIDE, was proposed, in which a root authority is responsible for establishing paths between hosts pertaining to different autonomous domains (ADs). The goal is to provide end-to-end data delivery in the presence of DDoS attacks. However, extra latency is incurred due to establishing an upstream path to the path server, as well as a downstream path from the path server to the destination.

2.3. *Traceback techniques*

They aim at filtering the attack packets as close to the attack source as possible, by reconstructing the actual path from the victim to the attacker, even in the presence of spoofed IP packets (Aljifri et al., 2003; Bai et al., 2004; Belenky and Ansari, 2007; Savage et al., 2000). For example Snoeren et al. (2001) present an audit trail method that uses a cryptographic hash function to generate lightweight packet digests stored by the network routers on the path, allowing to trace the origin of the attack source. Yaar et al. proposed the *Path identification* (Pi) technique (Yaar et al., 2003) that identifies the path taken by attack packets thanks to a mark inserted by the traversed routers. Due to the fact that different paths may end up with the same Path ID, the false positive rate is very high. Also, routers need to keep memory of the marks to be inserted.

2.4. *Cooperative detection and reaction*

These particular schemes rely on the collaboration between different communities or administrative domains that usually have *a prior* trust relationship. In these domains, security information or detection reports are periodically exchanged for achieving a broader detection coverage and a more reliable reaction. For example, a Cooperative Detection and Reaction Architecture (CDR) was presented in Koutepas et al. (2004). In CDR, every participating domain is assigned a Cooperative Counter-DDoS Entity, which is a modular software platform that automatically exchange security information in the form of alerts and heartbeat messages. Clearly, the effectiveness of this architecture depends on the extent of the participation from the different domains. Further, the multicast router is potentially vulnerable to DDoS attacks which may halt the exchange of the security information between the partners. A similar architecture called CITRA was reported in Schnakengerg et al. (2001). The independent security components (e.g., IDS, firewalls, routers) send attack reports to their CITRA neighbors to trace the attack path and initiate the response. CoDef (Lee et al., 2013) is a collaborative defense mechanism against flooding attacks, enabling autonomous domains to collaborate in order to redirect the legitimate traffic of customers upon request. This mechanism depends on a wide collaboration between different domains, and it does not block attack traffic in the source autonomous domain. Our design, *ArOMA*, shares a similar design purpose with CDR (Koutepas et al., 2004) and CITRA (Schnakengerg et al., 2001), but the assumption about trust domains and their collaboration does not necessarily hold. Also, instead of using multicast to communicate with different trusted domains, *ArOMA* simply exposes a security API that enables customers to leverage the programmability offered by the SDN controller.

2.5. *Traffic engineering*

In Stone (2000), the author proposed CenterTrack, an overlay-based architecture, in which special tracking routers are attached to the edge routers, and IP tunnels are pre-established from the edge routers to the special tracking routers, in order to detour the identified malicious traffic. Such overlay network clearly introduces operational complexity and additional administrative tasks. In Mahimkar et al. (2007), dFence was proposed to redirect suspicious traffic to middleboxes deployed in the core of the ISP network when a customer experiences congestion. One potential issue is that the suspicious traffic is stirred through a static sequence of middleboxes, some of which are not relevant to process the suspicious traffic, incurring more delay. In *ArOMA*, the potential latency is mainly caused by the communication between the SDN controllers, whereas customers do not experience such latency.

2.6. *SDN-based DDoS mitigation*

Recent years have witnessed much effort on developing novel cyber defense techniques that leverage the network programmability offered by SDN. For example, Li et al. have proposed Drawbridge (Li et al., 2014a), which allows end hosts within an ISP to subscribe to traffic engineering services provided by the ISP. It also requires ISPs to collaborate to perform traffic engineering. Therefore, the ISPs need to share their policy enforcement points (PEPs) with their customers, since the rules specified by the customers should be directly installed in the SDN switches of the ISPs. Different from Drawbridge, *ArOMA* allows the ISP to enforce the mitigation policy according to the security alerts sent by the customer. More recently, an SDN-based DDoS defense mechanism, called Bohatei (Fayaz et al., 2015), was proposed to support ISPs in providing DDoS defense as a service to their customers. Unlike our framework *ArOMA*, Bohatei requires the ISP to perform both the detection and the mitigation, which brings an important computational burden to the ISP, as it has to analyze a huge amount of traffic for different customer networks. Instead, *ArOMA* allows customers to run their local detection engines and only report alerts of concern to the ISP. In addition, Shin et al. developed FRESCO (Shin et al., 2013), a framework that enables security services to be customized and composed through the SDN controller, facilitating the deployment of security services, including DDoS mitigation schemes. However, the objective of our framework *ArOMA* is to provide collaborative DDoS mitigation between an ISP and their customers.

3. Key observations and design objectives

We analyzed and compared the state-of-the-art DDoS mitigation mechanisms discussed in Section 2, and summarized our findings in Table 1. A number of observations is given in the following.

- Most of the mechanisms in our sample are designed to protect end hosts. This is not surprising, considering the nature of most DDoS attacks, in particular recent occurrences such as DNS and NTP amplifications, or redirection attacks (Arbor Networks). The target link flooding attacks (Studer and Perrig, 2009), however, should not be omitted.
- Threat information exchange is not necessarily applied, depending on the design principle and architecture of the mechanism. Usually, the threat information exchange relies on either dedicated software and hardware add-ons (e.g., congestion-based mechanisms), or the modification of original routing devices (e.g., traceback techniques). Sometimes both of them are necessary (e.g., cooperative detection and reaction mechanisms). They may all lead to additional operational complexity, as well as computing and communication overheads, as most of them need to be operational continuously.
- One of the desirable features is that the DDoS mitigation scope should be *dynamically and optimally* distributed, in order to make the countermeasures more effective and reduce collateral damage. While some of the existing ones have this

feature to some extent, they still lack sufficient dynamicity and optimality.

- Most of the mechanisms rely on the manual configuration by the network administrator, as the enforcement of the mitigation policies are static.

With these key observations in mind, we are motivated to design such a DDoS mitigation mechanism that can overcome the identified limitations and achieve a number of desired properties, as follows:

- Broad protection coverage: protecting both end-host networks (e.g., end-users, CDNs) and communication links between ISPs and customers.
- Easy deployment: the modification of routing devices or the installation of special-purpose software or hardware are unnecessary. The threat information between the ISP and its customers is exchanged via a well-defined security API.
- Dynamic and optimized mitigation: ensuring QoS of victim customers in the presence of DDoS attacks, while avoiding collateral damage to other customers.
- Automated policy enforcement: achieving dynamic policy deployment through event-based reaction to security alerts and network state changes, alleviating the burden of human operators.

Thanks to the aforementioned properties, the security functions ranging from detection to reaction can be systematically integrated together, thereby reducing the latency of DDoS mitigation.

4. *ArOMA*: towards SDN-based autonomic DDoS mitigation

As Table 1 shows, most of the existing DDoS mitigation mechanisms perform well with particular assumptions, but they generally lack an effective way facilitating the collaboration between multiple parties, e.g., ISP, customer, as well as the systematic integration between different functional components, e.g., monitoring, detection, and reaction. In this Section, we present our autonomic DDoS mitigation framework *ArOMA* by leveraging the characteristics of SDN. We first present an overview of the architecture, along the operational workflow of *ArOMA*. We then specifically introduce its key components.

4.1. *Design architecture and operational workflow*

In the *ArOMA* framework, both the ISP and customers have their own SDN controller running in their respective network and communicating in a secure manner. As shown in Fig. 1, the framework is distributed across the ISP and customer networks, and the operational workflow (labeled with step numbers) can be described as follows:

1. As the traffic enters the ISP network, the controller assigns a unique flow identifier, `FlowID`, for each new flow, typically by leveraging the `PACKET-IN` message. `FlowID` is used by the customer controller when requesting for mitigation on this particular flow from the ISP. Labels are also assigned to flows by the ISP ingress switches for fast forwarding. The label helps preserving the policy applied to the flow to ensure unaltered routing.
2. On the customer side, flow statistics are periodically collected via OpenFlow agents, and forwarded to the detection engine for processing.
3. The detection engine relies on an attack database, which can interact with external databases via certain threat information exchange methods like the n6 API (Kijewski and Pawliński, 2014).
4. The detection engine performs any chosen type of network intrusion detection on the traffic received by the customer network. It generates security alerts in the presence of malicious and suspicious traffic flows, and triggers reaction from a local policy engine, which is in charge of generating and installing policy rules at the ingress switch for traffic processing.
5. The `FlowID`s of suspicious and malicious flows are encapsulated into a security alert that will be consumed by the mitigation engine residing on top of the SDN controller at the ISP side. To that end, a security API is exposed by the

Table 1 – Comparison between existing DDoS mitigation schemes.

Mitigation scheme	Detection point	Threat information exchange	Collaboration	Mitigation scope	Dedicated software and hardware	Modification of original routing device
Capability-based (Liu et al., 2010; Yaar et al., 2004; Yang et al., 2005)	End host	Not addressed	No	Single Site	No	Yes
Congestion control (Hsiao et al., 2013; Ioannidis and Bellovin, 2002)	End host	Addressed	Yes	Single Site	Yes	No
Traceback (Aljifri et al., 2003; Bai et al., 2004; Belenky and Ansari, 2007; Savage et al., 2000; Snoeren et al., 2001; Yaar et al., 2003)	End host	Addressed	Yes	Distributed	No	Yes
Cooperative Detection and Reaction (Koutepas et al., 2004; Lee et al., 2013; Schnakengerg et al., 2001)	End host	Addressed	Yes	Distributed	Yes	Yes
Traffic Engineering (Mahimkar et al., 2007; Stone, 2000)	End host	Not addressed	No	Distributed	Yes	No
ArOMA	End host	Addressed	Yes	Distributed	No	No

ISP's SDN controller that processes alerts issued by customers. This security API is the main component of the controller-to-controller communication.

6. Based on the alerts, the mitigation engine requests the policy engine for mitigation actions to handle the detected flows.
7. The mitigation engine further requests the *path lookup* module to infer the best path to enforce the mitigation actions on the flows of concern. The low-level rules corresponding to the mitigation actions are then distributed to the forwarding devices on the path returned by the path lookup module.
8. Mitigation actions (e.g, traffic redirection to security middleboxes) are finally enforced by updating the labels of the flows of interest to the label corresponding to the mitigation policy. This mitigation process involves automated and dynamic deployment of policies within the ISP network.

In practice, the operational workflow requires a few engineering assumptions and tackles a number of technical challenges. First, we assume that DDoS mitigation is provided as an on-demand service to the customers, who need to subscribe to the ISP beforehand. Second, the controllers of the customer and the ISP are assumed to be immune from attacks and are authenticated between each other. Finally, we suppose that the DDoS detection engine runs in the customer network and generates security alerts used for mitigation. More importantly, several major challenges need to be carefully tackled:

- *Scalable flow management:* The reactive nature of SDN makes it difficult for an ISP to provide DDoS mitigation as a service to their customers. Every time an OpenFlow switch receives a flow for which it does not have the entry in its flow table, it forwards that flow to the SDN controller to get the forwarding policy. Since this request always causes processing overhead in the SDN controller, a large amount of customers can make it hard for the ISP to handle the mitigation services in a timely manner.
- *Controller-to-controller communication:* To timely share the threat information between the customer and the ISP, the reliable and efficient communications between the SDN controllers of the ISP and customer networks plays a vital role.
- *Consistent forwarding policy:* The modification of packet header information often occur due to the wide use of networking devices like NAT. Such modification may impact the way the packets are forwarded, potentially violating the ISP network's forwarding policy.
- *Dynamic deployment of policies:* Mitigation policies should be deployed in an automated way depending on the request from the customer without non-trivial human intervention or sophisticated manual configuration.

4.2. *Design components: their roles and working principles*

The essential design components of *ArOMA*, as shown in Fig. 1, are described below. We take a bottom-up approach by first covering the data plane components, and then the control plane components.

4.2.1. *Data plane components*

While the data plane is mostly comprised of packet-forwarding devices, mainly the OpenFlow switches, packet-processing functions supported by the middleboxes are also deployed in the network.

4.2.1.1. *OpenFlow switches.* According to the OpenFlow specifications (Open Networking Foundation, 2013), 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 a packet, OpenFlow switches perform a lookup operation in the flow table: if it does not have a flow entry for that packet, then it is considered the first packet of the flow, and its header information is forwarded to the controller using a PACKET-IN message.

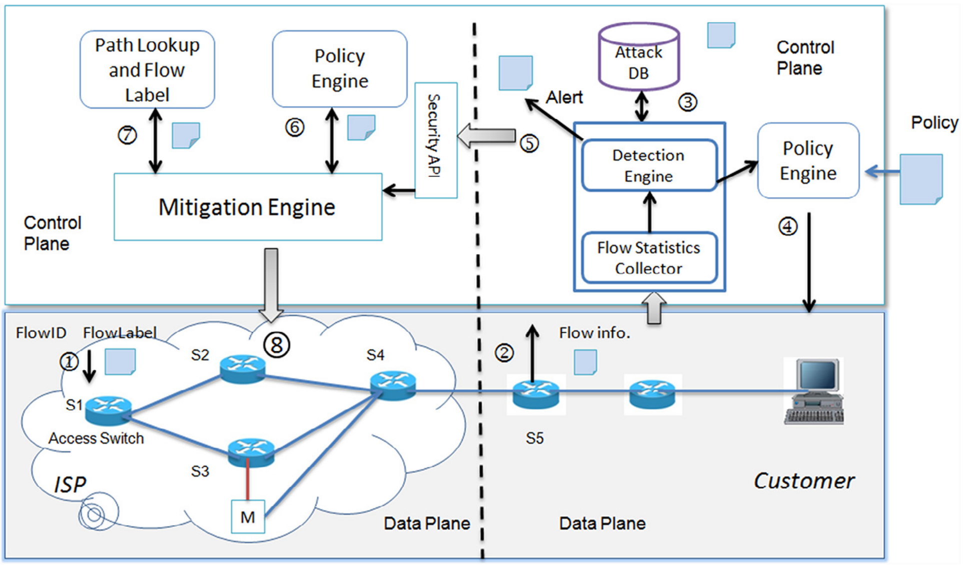

Fig. 1 – SDN-enabled DDoS mitigation framework.

4.2.1.2. Middlebox (M). The term, as used here, generally refers to the devices that host the security functions that are able to mitigate the DDoS attacks. These devices are deployed in the network for processing and filtering the suspicious and malicious traffic. As shown in Fig. 1, a middlebox *M* is deployed in the ISP network, along with switch S3. In our framework, we assume that *M* enforces pre-defined security policies for mitigating DDoS attacks, such as UDP, TCP SYN or ICMP floods. For instance, a middlebox monitors the source IP address of the suspicious flow which sends UDP packets. If the source sends a huge amount of packets (above a given threshold) then it is categorized as malicious and dropped, otherwise it is treated as legitimate. More sophisticated policies dealing with amplification and redirection attacks can be specified as well.

4.2.2. *Control plane components*

Components in the control plane span across the customer and the ISP networks to complete the autonomic operational loop, in which the customer provides traffic monitoring and detection, while the ISP provides mitigation countermeasures.

4.2.2.1. Customer side. **Monitoring Component.** It is composed of two modules to perform the task of collecting incoming traffic and detecting attacks in the customer network.

- *Flow Statistics Collector:* flow statistics are periodically requested by the customer controller from the OpenFlow switches and forwarded to the *detection engine* for further analysis and processing. Some flow statistics collection techniques have already been experimented with SDN technology in Akbar Mehdi et al. (2011) and Braga et al. (2010). In the proposed framework, the OpenFlow collector (OF collector) has been used for collecting the statistics.
- *Detection Engine:* taking flow statistics as inputs, it generates security alerts in the presence of anomalous traffic, based on common anomaly-detection algorithms or using signatures from the *attack database*. The alerts then trigger the local *policy engine* for incoming flows to be processed accordingly, as a first layer of defense. Proposing a new detection algorithm is outside the scope of this paper. However, for the implementation purpose, we assume that the customer network uses a detection technique to flag whether it is under attack or not based on traffic rate threshold (Fayaz et al., 2015; Mahimkar et al., 2007).

Attack Database. We assume such a database which contains previously-seen anomalous traffic and their associated signatures (Mazel et al., 2014). It is maintained by the customer network and possibly updated on a subscription basis at security vendors and other trusted third parties. The detection engine then compares the characteristics of anomalous flows with the flow features in the database to detect attacks. Especially, we envision that such a database can interact with other threat intelligence mechanisms like NECOMAtter (Iimura et al., 2014) using a common interface such as n6 (Kijewski and Pawliński, 2014).

Policy Engine. The policy engine enables the ISP and the customer to select and deploy security policies in their respective network. The policy engine at the customer controller generates the local rules to tackle the identified anomalous flows based on the alerts issued by the *detection engine*, and possibly characterized thanks to the *attack database*. Security alerts are formatted using IDMEF (Feinstein et al., 2015) (a description of the format is given in Section 5) and contain the source and destination IP addresses, as well as the flow identifier `FlowID`, a `Security_Class` which provides classification information on the detected flow, and the `Impact_Severity` of the detected flow on the customer network. `Impact_Severity` is categorized into three categories: `Low`, `Medium` and `High`. Depending on the information mentioned in the alert received from the customer controller, the *policy engine* at the ISP side enforces the policy to mitigate the attacks.

As for the `Security_Class`, we distinguish legitimate traffic from suspicious and malicious classes of traffic. The motivation is to minimize the impact of countermeasures applied on the flows in the ISP network by considering an intermediate class of traffic, *suspicious flows*, on which we cannot decide. Suspicious flows are not blocked in the ISP network, but may be redirected through paths with a lower quality of service. In the ISP network, different paths are provisioned from the ingress point to the the egress points (end-to-end) for different suspicious traffic based on its impact severity. Thus, we are able to reduce the collateral damages on the legitimate traffic, instead of blindly dropping the target flows. The mechanism to classify the flows is out of the scope of this paper, and we assume that the customer uses some detection mechanism to classify the flows as suspicious and malicious (Braga et al., 2010). Our focus is about enforcing the policy to mitigate the attacks once it is detected.

Listing 1: A Sample Policy File: Redirection of Suspicious Traffic

```
<Policy PolicyID="Security_policy">
    <Event attack="ICMP-Flood">
     </Event>
    <Condition>
        <flow class="suspicious"/>
        <Impact severity="low"/>
     </Condition>
    <Actions action="Low_Bandwidth_Path"/>
     </Actions>
</Policy>
```

Moreover, policies are defined as an event, condition and action paradigm (Damianou et al., 2001). Attack type is used to define the event in our policy file. Flow class and impact severity are used to specify the conditions of the policy. A high-level action such as drop or low_bandwidth_path are defined in the action part of the policy as shown in the Listing 1 and Listing 2. For example, if the alert specifies the security class as suspicious, impact severity as low, and attack type as ICMP-Flood then the policy engine uses these informations to check the policy file to enforce the action on the flow. As shown in the Listing 1, the policy specifies to provide the low bandwidth path to the flow with low impact severity and suspicious in nature. Similarly, if the security class is malicious then the high level action drop is specified as shown in the Listing 2, according to the policy at the ISP controller.

Listing 2: Mitigation of Malicious Traffic Policy

```
<Policy PolicyID="Security_policy">
    <Event attack="ICMP-Flood">
    </Event>
    <Condition>
        <flow class="malicious"/>
    </Condition>
    <Actions action="Drop"/>
    </Actions>
</Policy>
```

4.2.2.2. *ISP side*. **Security API.** We envision that the ISP provides DDoS mitigation as an on-demand service via a security API provided by its SDN controller. Through this security API, the customer can request the ISP to deploy middleboxes to filter suspicious and malicious flows, and to assign a higher priority to legitimate flows. In particular, this Security API is implemented at the ISP controller as a REST API, which receives mitigation requests from the customer controller. The detection engine at the customer controller sends security alert in the standard IDMEF message format. Upon receiving the alert, the Security API extracts the alert information and forwards the information to the mitigation engine at the ISP controller.

Such security service interface is being currently discussed at the IETF under the working group I2NSF (Dunbar et al., 2014) which aims at standardizing these security functions, prompting them to be available for widespread use in coming years. Moreover, the security API enables the customers and ISPs to share threat information with different ISPs and customers which enables *ArOMA* to be extended for multiple customers and multiple ISP scenarios. Table 2 shows the HTTP requests the security API handles with the specified parameters. These parameters are forwarded then to the policy engine, which finally takes the appropriate action.

Flow Label Module. To tackle the two challenges identified at the beginning of this Section, i.e., the *scalable flow management* and the *consistency of the forwarding policy*, *ArOMA* uses labels to identify the path from the ingress switch to the egress switch (end-to-end path) of the ISP network. Labels are used for fast switching and rerouting, as core switches in the ISP network simply need to check the label and forward the packets. Additionally, by using labels, the path policy is preserved through the network, as the modification of the packet header, e.g., by a NAT device, does not cause a `PACKET-IN` (Open Networking Foundation, 2013) event at subsequent switches, since only the label is checked for processing. Another advantage of using labels is to reduce the flow table entries in the core switches of the ISP network (Yeganeh et al., 2013), resolving the concern of scalability.

Table 2 – Security API overview.

Request	Arguments	Response
GET url/redirect	id = FlowID, Security Class, Impact Severity, attack type	status = 200 OK
GET url/block	id = FlowID, Security Class	status = 200 OK

Algorithm 1 $FlowLabel(packet) \rightarrow labeled\_packet$

```
for each incoming packet $P$ do
    if Packet is in the flow table then
        $forward(P, out\_port)$
    else
        Check the destination IP and destination port
        $P.VLANID \leftarrow Push(Legitimate\_label)$
        $forward(P, out\_port)$
    end if
end for
```

Algorithm 1 illustrates the flow label assignment process. Specifically, the label is inserted in the 12-bit VLAN ID field using the *Push* method of OpenFlow (Open Networking Foundation, 2013). At the ingress switch of the ISP, flow rules are pre-installed based on the destination IP address of the customer network. When a flow arrives in the ISP network, it is checked whether the flow is present in the flow table. If the flow table contains the flow then a label is assigned to the flow and it is forwarded through an output port. Otherwise, flow informations are parsed and its destination IP and port informations are checked and then a label is assigned to the flow, and it is forwarded through an output port of the ingress switch. As such, the switch-controller interactions to assign labels to flows can be significantly reduced, alleviating overhead on the controller as well.

The reason of using the VLAN ID field to assign labels is that it can provide $2^{12} = 4096$ different labels in its single domain, which are sufficient for a single domain of ISPs, which usually has a small number of links connected with their customers through its single domain, e.g., 500 links in Ebone (Spring et al., 2004; Wichtlhuber et al., 2015).

Path lookup. This module provides the ISP with the ability to maintain a list of end-to-end paths at its controller. Paths in the ISP network spanning from the ingress switch to the egress switch are computed using an all-pairs shortest path algorithm (Demetrescu and Italiano, 2003). In *ArOMA*, a pool of paths is maintained by the ISP according to different bandwidths, link loss probability and delay to accommodate different types of flows which are categorized into legitimate and suspicious ones. In doing so, the ISP can provide differentiated traffic engineering to its customers, as for example, suspicious flows would be transported to paths providing low QoS with a higher number of hops. On the contrary, legitimate flows would be flown along paths providing high QoS with a small number of hops. Specifically, it receives the input in the form of middleboxes to traverse or type of path (e.g., low bandwidth path) with impact severity (e.g., suspicious or legitimate) from the mitigation engine and returns any matching path and its associated label. It allows the ISP controller to redirect the flow through a policy-based path rather than broadcasting the traffic towards security devices (Mahimkar et al., 2007).

Flow Identifier. Aggregating the flows at the ingress switch of the ISP network protects these switches from flow table exhaustion. *ArOMA* uses the `FlowID` to identify and aggregate the flows at the ingress switch of the ISP network and share them between the ISP and the customer controllers (see Algorithm 2 for `FlowID` generation). Specifically, a `FlowID` is computed using the IP-4 tuple (source IP, destination IP, source port, destination port) as input to the SHA1 algorithm. Flows from the same source to the same destination are aggregated under the same identifier. The `FlowID` is inserted into the 64-bit cookie field set by the controller and is stored at the ingress switch of the ISP network. When the flow reaches the customer network, then its controller requests the ISP controller for the corresponding `FlowID` by sending the source and destination IP addresses of the flow.

Additionally, it protects the ingress switches from state exhaustion attacks due to the limited TCAM entries, however we still need to maintain the flow states at the ingress switch. As the size of flow tables is increasing to meet the requirements of SDN deployments, we are now able to use software switches (e.g., OVS switches) (widely used in the datacenters as well) to avoid the forwarding state limitation of hardware switches (Kreutz et al., 2015). Moreover, switching devices with high performances chips (e.g, EZchip NP-4) can provide optimized TCAM memory that supports from 125,000 up to 1,000,000 flow table entries (Kreutz et al., 2015).

Algorithm 2 `FlowID`$(flow) \rightarrow flow\_identifier$

```
for each incoming flow $f$ do
    $f \leftarrow Flow(f.IP\_src, f.Port\_src, f.IP\_dst, f.Port\_dst)$
    if $f.IP\_dst \in Ingress\_switch.FlowTable$ then
        $f.cookie \leftarrow$ FlowID
        $forward(f, out\_port)$
    else
        $controller(f)$ // Flow is forwarded to controller for FlowID
        $new\_$FlowID $\leftarrow hash(f)$ // new FlowID is assigned
        $f.Cookie \leftarrow Push(new\_$FlowID$)$ // the new FlowID is inserted in the Cookie field
    end if
end for
```

Mitigation Engine. This module communicates with other modules deployed at the application layer of the ISP controller in order to enforce the final mitigation countermeasures. Upon receiving information extracted from the alert message (i.e., `FlowID`, `Security_Class`, `Impact_Severity`, source IP, destination IP) at the Security API, it forwards these informations to the *policy engine* at the ISP side, which provides high-level policies (e.g., Redirect, Drop) to be enforced. When it receives a high-level action from the policy engine, the mitigation engine also checks the *Path lookup* module for getting the appropriate path and the associated label to be assigned to the flow. For simplicity, we assign labels 1 and 2 to the legitimate flows and 3 to 6 to suspicious flows respectively with different impact severities.

For instance, if the security class and impact severity specified for the flow is suspicious and high respectively, then the high-level action is provided as to redirect the flow through the lowest bandwidth path. Then, the mitigation engine checks the *Path lookup* module to get the label associated with the lowest bandwidth path and modifies the label of the corresponding flow at the ingress switch to be redirected. Similarly, if the flow is malicious and the high level action is *drop*, the mitigation engine modifies the action for the corresponding flow to be blocked at the ingress switch of the ISP.

4.3. *Controller to controller communication*

Another important feature of *ArOMA* is the communication between the controller of the ISP network and the one of the customer network. Controller-to-controller communication enables the customer to share the alert information with the ISP in an automated way. This communication channel actually serves two purposes: sharing the `FlowID` from the ISP to the customer, and alerting the ISP about suspicious and malicious traffics incoming the customer network. Specifically, the way for sharing `FlowID` is described as follows:

1. Initially, when the flow reaches the customer network, the customer controller requests for the `FlowID` of the corresponding flow by sending the IP addresses (source IP, destination IP) of the flow to the Security API at the ISP controller.
2. The `FlowID` is maintained and periodically updated at the ingress switch of the ISP in the cookie field.
3. On receiving the request from the customer controller, the ISP controller checks its ingress switch for the `FlowID` and sends it back to the customer controller.

For example, the controller communication for DDoS mitigation mainly contains the following operations:

1. Upon detecting DDoS traffic at the customer side, the detection engine sends the alert for the detected flows;
2. The security alert is sent using IDMEF (different formats can be used as well).
3. The Security interface (Security API) deployed at the ISP controller is invoked via a REST API by sending the security alert message through the detection engine at the customer controller.
4. The Security API extracts informations such as, the security class, flow informations (source IP, destination IP), impact severity and attack type from the security alert.
5. An acknowledgement message containing the flow information and action enforced is returned to the customer controller after action is enforced on the flow.

5. Use case

For better illustration and understanding of the design purpose and principle of *ArOMA*, we develop a use case, as shown in Fig. 2, in which one customer and one ISP collaborate with each other to mitigate DDoS attacks. A use case including multiple customers and multiple ISPs will be studied as our future work. In addition, for a single ISP network, it is possible to deploy multiple SDN controllers, each of which interact with one customer network through the security API.

First of all, the customers that intend to run *ArOMA* need to subscribe to their ISP for this security service, and obtain authentication credentials (Step 0 and Step 1 in Fig. 2). Meanwhile, an anomaly detection system needs to be set up and configured in the customer network, in order to monitor and detect the unwanted traffic according to its local security policies. It is worth noting that neither dedicated hardware nor special software is required for deploying *ArOMA* in the customer network, except for the fact that the network should be SDN-enabled. This yields significant flexibility for the customers, as they are allowed to run different, even multiple, detection engines and develop specific attack characterization by taking into account their local context, policies and requirements.

For Step 2, we assume a threat information exchange format is well defined between the ISP and customer controllers, so that the alert information can be issued to the ISP as soon as an attack of concern is detected in the customer network. To enable this process, the ISP deploys a security function via a RESTful API at its SDN controller to handle the request received from the customers. Upon receipt of the alert, the security interface verifies its validity first, and then extracts the information from the alert and forwards it to the policy engine. Then, the policy engine checks whether a policy matches the received alert and enforces the corresponding rules in its SDN switches. For example, the IDMEF format can be employed for our purpose thanks to its flexibility and compatibility with other formats. An example alert is shown in Listing 3, indicating suspicious traffic with a low impact severity. Specifically, the *Classification* and *Assessment* fields specify the type and impact of the attack respectively. The latter indicates the *impact severity* of the detected traffic with a qualitative value among {low, medium, high}. Apart from the IP addresses of the attack source and victim network, two additional fields, the *Security Class* and `FlowID` are specified in the alert as well.

It is worth noting that the policy engine needs to take into account the status of the ISP network, ensuring the legitimate flows are always assigned routing paths with high QoS, and the collateral damage on other customer networks can be

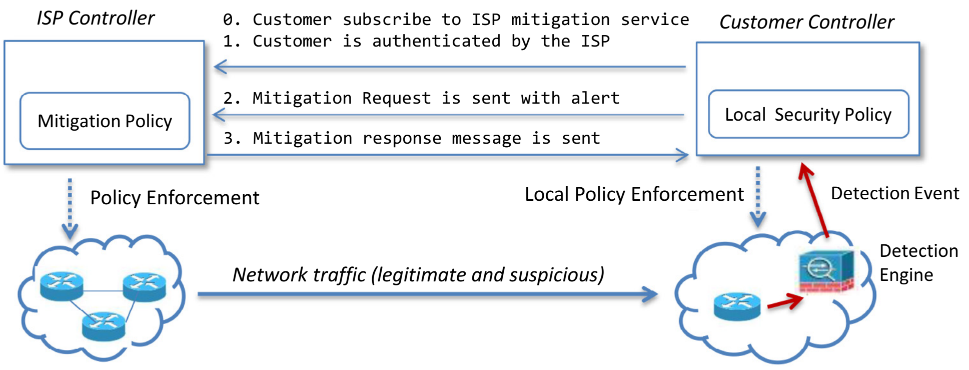
ISP Controller
Mitigation Policy
0. Customer subscribe to ISP mitigation service
1. Customer is authenticated by the ISP
2. Mitigation Request is sent with alert
3. Mitigation response message is sent
Customer Controller
Local Security Policy
Policy Enforcement
Local Policy Enforcement
Detection Event
Detection Engine
Network traffic (legitimate and suspicious)

Fig. 2 – Use case illustrating the application of *ArOMA*.

reduced. A path lookup algorithm can be run here for this purpose. As a result, the ISP notifies the customer about the mitigation process, indicated as Step 3. To complete this process, it is even possible to have an extra step in acknowledging the status of the customer network after mitigation.

As Fig. 2 shows, although our current prototype is developed for the ISP networks, it can be easily implemented and deployed in data center networks as well, in which we assume that security middleboxes can be dynamically created and deployed via network functions virtualization (NFV), for example. From an architectural perspective, the operational loop is closed, ranging from traffic monitoring and anomaly detection (customer side) to traffic engineering and attack mitigation (ISP side).

Listing 3: Threat Information Exchange Format Between ISP and Customer Network

```
<IDMEF–Message version=”1.0”>
<Alert>
    <Analyzer analyzerid=”CUSTOMER_C”/>
    <Source>
        <Node>
                <Address category=”ipv4–addr”>
                        <address >10.0.0.2</address>
                </Address>
                </Node>
    </Source>
    <Target>
        <Node>
                <Address category=”ipv4–addr”>
                        <address >10.0.0.3</address>
                </Address>
                </Node>
    </Target>
    <Classification attack=”UDP–Flood_Attack”>
    </Classification>
    <Assessment>
        <Impact severity=”low”/>
    </Assessment>
    <AdditionalData> type =”string” meaning=”security_class”>
        <string>suspicious</string>
    </AdditionalData>
    <AdditionalData> type =”string” meaning=”FlowID”>
        <string >196764794928656</string>
    </AdditionalData>
</Alert>
</IDMEF–Message>
```

6. Simulations

The purpose of our experiments is to validate the feasibility and effectiveness of *ArOMA*, our proposed autonomic DDoS mitigation framework. We validate the prototype using both mininet-based simulations and testbed-based experiments. We first describe the threat model that is addressed by *ArOMA*, we then present the detailed settings and configurations of our experiments, including the implementation details of the prototype.

6.1. *Threat models*

We are concerned with DDoS attacks against the customer of an ISP. Attackers may be distributed and can attack through the ingress routers in the ISP network to flood the ISP and customer network. Our current work is focused on a single customer network, while the multiple customers scenario will be studied in the future work. Also, we restrict our focus to DDoS flooding attacks. In particular, two common DDoS flooding attacks, which cause collateral damage to the ISP and its other customers, are described below.

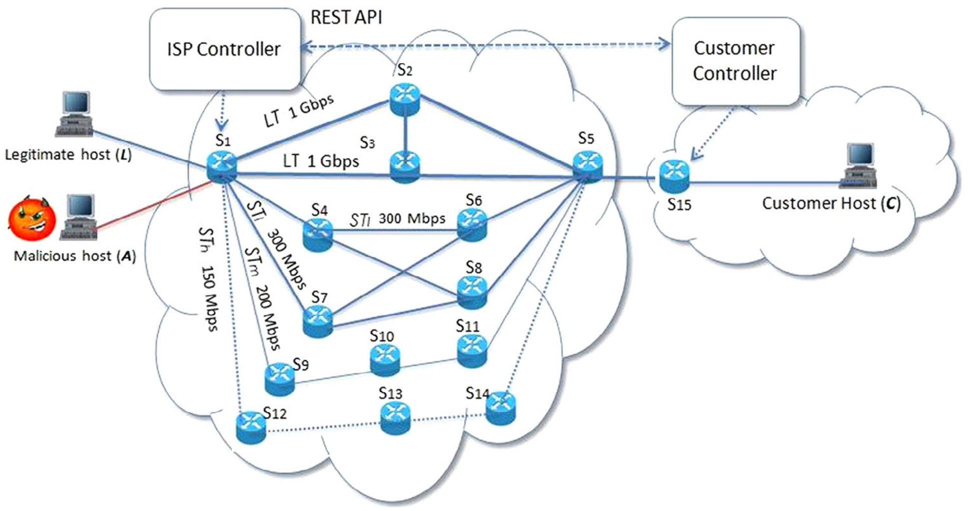

Fig. 3 – Network topology used for the simulations.

1. *Target flooding attacks*: an adversary sends a huge amount of traffic to the customer network to rattle up the customer's communication. Generally, an adversary could use UDP flood, TCP SYN flood or ICMP flood attacks to disrupt the services provided by the customer network.
2. *Bandwidth flooding attacks*: the adversary's objective is to flood the link between the ISP and customer networks, ultimately disrupting the legitimate traffic which share the same link. These types of attacks causes collateral damage to other customers of the ISP, as it floods the links of the ISP. UDP flood and ICMP flood are widely used for launching such attacks.

6.2. *Network topology*

The prototype is implemented in Python and runs as an OpenFlow application on the Ryu controller. To perform the simulation, we employ mininet, which provides rapid prototyping environment for the emulation of OpenFlow switches (Lantz et al., 2010). In particular, we emulate two OpenFlow networks as shown in Fig. 3, and the traffic is generated by using iPerf (Qin et al., 2003). We use UDP flood as the attack traffic in this simulation.

We consider the network topology of the ISP and its customer as shown in Fig. 3, which has 14 OpenFlow switches, noted as $S_1, S_2 \cdots, S_{14}$, connected with the ISP controller.

In particular, the ingress switch S_1 is connected to the external network, and the ISP controller applies the policies on the flows coming from the external network at S_1 (at least the process of labeling packets). Switches in the ISP network are controlled by the ISP controller, and the switches in the customer network are controlled by the customer network. Moreover, we assume that those communication links in the ISP network provide different capacities:

- links reserved for legitimate traffic (*LT*) provide 1 Gbps capacity;
- links reserved for suspicious traffic with a low impact severity (ST_l) provide 300 Mbps across 4 hops;
- links reserved for suspicious traffic with a medium impact severity (ST_m) and a high impact severity (ST_h) are configured with link capacities of 200 Mbps and 150 Mbps, respectively, and are 5-hops long.

6.3. *Simulation scenario*

Our simulation scenario involves all the components described in Section 4.2, as shown in Fig. 3. In particular, two external hosts, one legitimate (denoted as L) and one malicious (A) communicate with a host (C) in the customer network. SDN controllers in the ISP and the customer networks communicate with each other via REST APIs.

We assume that the ISP maintains a policy table as shown in Table 3, and the ISP controller installs the labels beforehand in the flow table of the ingress switch S_1. Then the traffic between the two external hosts and host (C) is labeled with

Table 3 – Mapping between security class, impact severity, policy, bandwidth and associated label.

Security class	Impact severity	Policy	Bandwidth	Path	Label (VLAN ID)
Legitimate	Empty	Forward	1 Gbps	(S_1, S_2, S_5)	1
Legitimate	Empty	Forward	1 Gbps	(S_1, S_3, S_5)	2
Suspicious	Low	Redirect	300 Mbps	(S_1, S_4, S_6, S_5)	3
Suspicious	Low	Redirect	300 Mbps	(S_1, S_7, S_8, S_5)	4
Suspicious	Medium	Redirect	200 Mbps	(S_1, S_9, S_{10}, S_{11}, S_5)	5
Suspicious	High	Redirect	150 Mbps	(S_1, S_{12}, S_{13}, S_{14}, S_5)	6

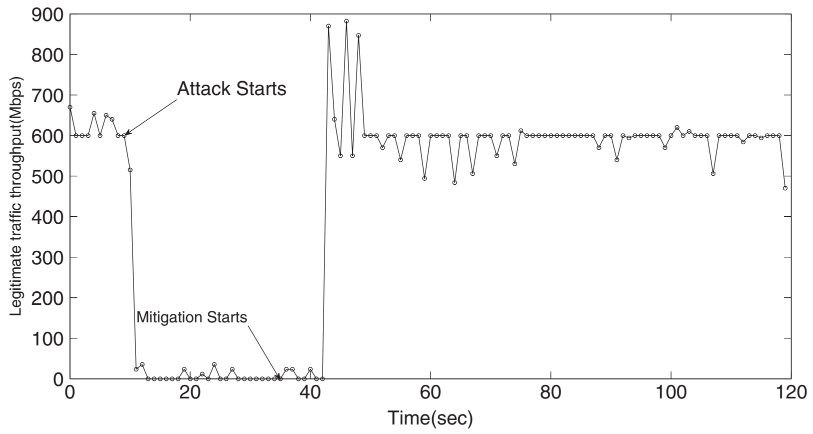

Fig. 4 – Response of ArOMA and throughput of legitimate traffic.

the respective `FlowID`s as it enters the ISP network. Upon receiving new flows, i.e., flows for which the controller does not have any forwarding policies, the ISP controller considers them as *legitimate* and assigns label "1" or "2" (determined by a load balancing scheme) (Li et al., 2014b). As a result, all the intermediate switches simply check the label in the VLAN ID field and forward the flows through legitimate path. Finally, the egress switch S_5 pops the label "1" or "2" from the flows then forwards them to the customer network.

When the ISP controller receives an alert from the customer controller, it modifies the label accordingly. In reaction to the example alert shown in Listing 3, ISP controller will enforce the *redirect* policy by inserting label "3" to all of the flow packets originating from host `A`. Meanwhile, legitimate flows with label "1" originating from host L are still forwarded through the path with a high QoS.

6.4. *Simulation results*

We conducted simulations to evaluate the end-to-end effectiveness, throughput and network jitter of the legitimate traffic. We also evaluated the throughput of suspicious traffic when they were processed according to their impact severity. Formally, the following two metrics are carefully studied,

- Throughput and network jitter of legitimate traffic in case the presence of attacks;
- Throughput of suspicious traffic under different impact severity;

6.4.1. *End-to-end effectiveness*

We evaluated the effectiveness of our prototype *ArOMA* in how quickly it can restore the throughput of legitimate traffic in the presence of given DDoS attacks. The results are shown in Fig. 4. Specifically, we launched the UDP attack traffic at the 10th second. Then upon receiving the mitigation request from the customer controller, the ISP controller started the mitigation at around the 35th second. As clearly indicated in the figure, it took less than 10 seconds for the legitimate traffic to recover its throughput after the mitigation started. Thus, we can conclude that *ArOMA* generally provides rapid mitigation response in restoring the performance of the legitimate traffic.

6.4.2. *Network jitter*

We measure the network jitter of the legitimate traffic for determining the variation in arrival time between packets, and further understanding the impact of DDoS attack on the quality of service of the legitimate traffic. As shown in Fig. 5, the attack traffic is launched at the 10th second, then immediately the

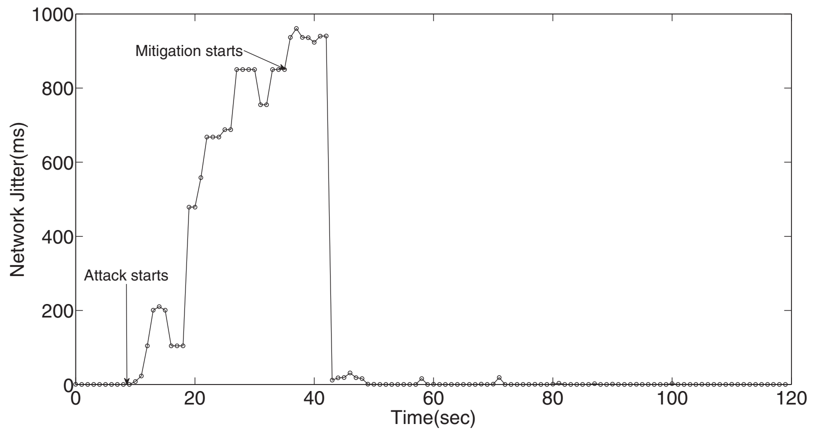

Fig. 5 – Network Jitter of legitimate traffic going towards customer network.

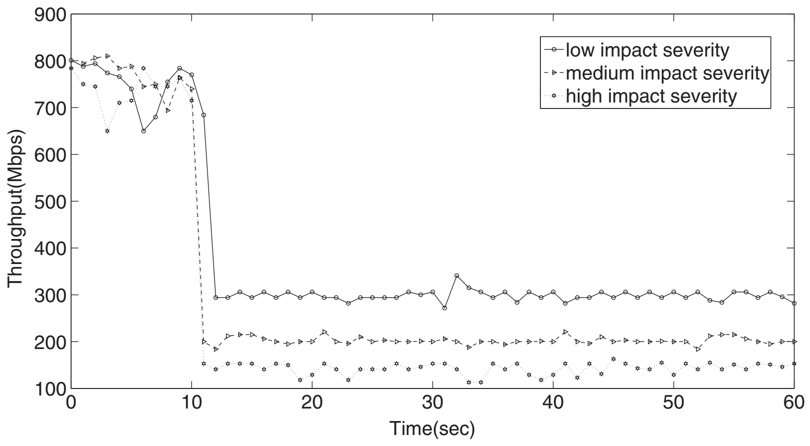

Fig. 6 – Throughput of suspicious traffic with different impact severity.

network jitter started to increase. Clearly, when the mitigation was activated upon receiving the alert close to 40 seconds from the customer, the network jitter decreased to around 0.5 ms within 10 seconds.

6.4.3. *Throughput of suspicious traffic*

Another interesting experiment was to test the throughput of the suspicious traffic, which is classified into three types in terms of its impact severity on the customer network: low, medium and high, as shown in Table 3. In the ISP network, suspicious traffic is not dropped as it may be legitimate. So, to avoid collateral damage to these kinds of traffic, the ISP provides them different processing through path with more number of hops and less bandwidth. As the results shown in Fig. 6, the throughput of the suspicious traffic with *low* impact severity was maintained at around 300 Mbps, while the suspicious traffic with *medium* and *high* impact severity was redirected through the path with larger number of hops, with throughputs close to 200 Mbps and 150 Mbps respectively.

7. Testbed based experiments

In addition to the simulation based experiments, we also experimented with the actual platform to evaluate our prototype *ArOMA*. In the platform, we treat the video streaming service as the target to protect, and measure its quality, in the face of DDoS attacks, in terms of Quality of user Experience (QoE) metrics.

7.1. *Experimental settings*

7.1.1. *Platform*

The topology of our experimental platform is similar to the data plane configuration shown in Fig. 1, where the ISP and the customer networks have their own SDN controllers. The components and specifications of the platform are given in Table 4. For simplicity, two routing paths are configured in the platform for the ISP network: one is QoS guaranteed and used exclusively for legitimate traffic (going through switches S_1, S_2, and S_4), while the other is for suspicious or malicious traffic (going through S_1, S_3 and S_4). In the customer network, the OpenFlow switch S_5 is attached to the customer controller. Also, ten host machines are virtualized in the platform, serving as legitimate host, attack host and customer machine.

7.1.2. *Traffic generation*

We use video as the source of legitimate traffic, considering the fact that nowadays video traffic accounts for more than 70 percent of all consumer Internet traffic (Dobrian et al., 2011a). We use *TAPAS* (De Cicco et al., 2014), a video streaming tool for video traffic generation between the customer machine C and the legitimate host *L*. 10 *TAPAS* video clients generate a legitimate video traffic over HTTP traffic. Moreover, the BotNet

Table 4 – Platform specifications.

Component	Quantity	Specification or configuration	Role
IBM RackSwitch G8052	5	48 Gigabit ports	OF enabled switch
DELL server r620	2	8-core 2.9 GHz CPU, 16 GB RAM, 10 Gb/s Ethernet NICs	SDN controller host (ISP, customer)
Hosts	3	Ubuntu server 12.04.2, 6-core 3 GHz CPU, 4 GB RAM, 20 GB HDD	Playing as video streaming server *L*, attack host A and video streaming client C
SDN Controller	2	Ryu 3.20	Controlling switches in ISP (S_1 to S_4) and customer networks (S_5)

Table 5 – Defined metrics to measure the impact of DDoS attacks.

Metric	Definition	Unit	Impact of attack
Time to rebuffer	the time taken to rebuffer the video when the streaming buffer gets empty due to network congestion	second	increases
Time to start	the time taken to complete the initial buffering to start the video streaming	second	increases
Duration of paused mode	the time player is in the paused mode for the buffer to fill up during the entire video length	second	Paused for more time
Average buffer length	the buffer length of the player	second	increases
Average goodput	rate of data transfer	KB/sec	decreases

Simulator (BoNeSi) tool is used to generate volumetric DDoS attacks with high traffic rates. We generate an attack traffic close to 250,000 packets per second.

7.1.3. *Evaluation metrics*

The metrics used to evaluate our prototype are specified in Table 5, which are essentially related to these QoE metrics analyzed in Dobrian et al. (2011b), Krishnan and Sitaraman (2013) and Seufert et al. (2015). Our hypothesis is that in the presence of DDoS attack, *time to rebuffer* will increase, *average buffer length* will decrease and the *duration of player in the paused mode* will increase. The intuition behind is that attack traffic may deplete the playout buffer of the video player, eventually reducing the quality of user experiences watching the video. Our experiments will test the hypothesis.

7.2. *Results and analysis*

We conducted several rounds of tests to comparatively study the given metrics in three cases,

- Video streaming services in normal condition (without attack traffic);
- Under different attacks without mitigation;
- And under attack with our prototype *ArOMA* activated.

The metrics are measured at the video streaming client by collecting and analyzing the log files generated by *TAPAS*.

7.2.1. *Time to rebuffer*

The hypothesis about this metric is that the client is not able to play the video as the streaming buffer gets depleted due to overwhelming attack traffic. If such a condition holds for a long time, the QoE of users will be degraded. Fig. 7 clearly validates this hypothesis: the *TAPAS* video client is able to complete the video transmission without any interruptions when there is no DDoS traffic, and the buffer length is maintained above 15 seconds. In contrast, under DDoS attack, the video client was not able to buffer the video, leading to a depletion of the playout buffer down to zero. When the mitigation scheme is activated by the ISP upon the request of the customer controller, the playout buffer returns to a normal level, allowing the video client to play the video. In the case of TCP and ICMP flood it takes more time to start the buffering, since because of congestion *TAPAS* video client is disconnected from the server. When the buffering started, it took 10 to 15 seconds for the playout buffer to return to the normal level.

7.2.2. *Average buffer length*

We measured the average buffer length, and observed that the buffer length is maintained at 16 seconds when there is no attack. In the presence of UDP flood attack, the buffer length is maintained close to 14 seconds with *ArOMA* activated. Similarly, the buffer length is maintained around 12 and close to 10 seconds, in face of ICMP and TCP flood attacks respectively. In contrast, the average buffer length was about 4 seconds when the mitigation module is not activated during attacks (Fig. 8).

7.2.3. *Duration of paused mode*

This metric is evaluated for comparing how much time the player is in the paused mode during the initial buffering and in the midst of the attack. The results are shown in Fig. 9, which has the following implications,

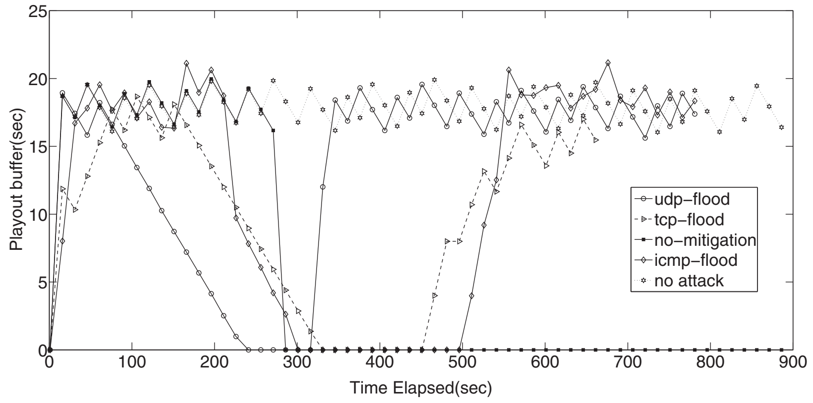

Fig. 7 – Time to rebuffer.

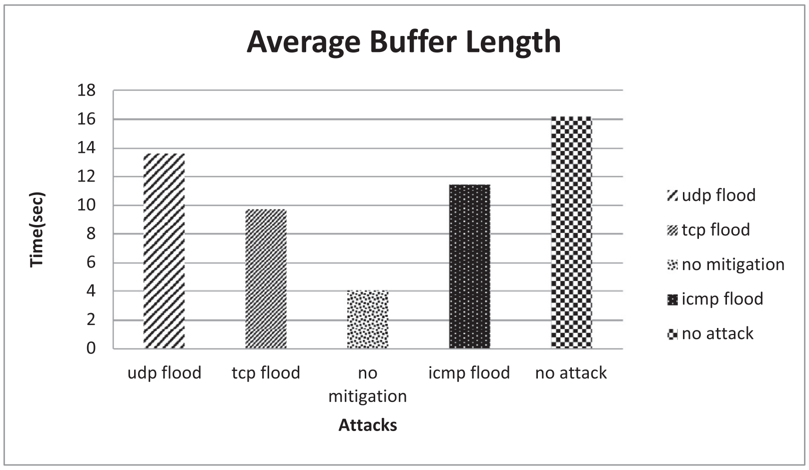

Fig. 8 – Average buffer length of legitimate and attack traffic.

- When there is no attack, the total duration that the player is in the paused mode is approximately 2 seconds during the initial buffering phase;
- With *ArOMA* activated, the video player is paused for 12 seconds under UDP flood attack. The values are close to 20 and more than 40 seconds under TCP SYN flood and ICMP flood attack respectively. Since the video client is disconnected from the server in case of TCP and ICMP flood attack;
- Noticeably, the player is not able to buffer the video to play if *ArOMA* is not activated.

7.2.4. *Average goodput*

We also examined the *average goodput*, and observed it is maintained around 600 KB/sec in the absence of attacks. When DDoS attack is launched and *ArOMA* is activated, the average goodput could be maintained at 550 KB/sec against the UDP flood attack. As for ICMP and TCP SYN flood attacks, it is maintained close to 500 and 450 KB/sec respectively, as indicated in Fig. 10. In contrast, when *ArOMA* was not activated, the average goodput dropped down below 300 KB/sec and could not return to a normal level.

7.3. *Discussion*

Both the simulation based and testbed based experimental results demonstrated that *ArOMA* is able to provide quick response in mitigating attack traffic. It is worth noting, however, classifying the flows as suspicious and malicious are out of concern of our proposed framework. We assume that the customers can deploy some detection mechanisms to detect and classify the flows as suspicious and malicious based on their local intelligence and criteria. Nevertheless, *ArOMA* provides a distributed and automated way to distinguish suspicious traffic in terms of their severity, i.e., they are assigned with different labels and redirected to different paths. Moreover, although the attack traffic used in the experiments are mainly UDP-flood, TCP-flood and ICMP-flood, *ArOMA* can also be applied to mitigate other types of attacks as long as the victim customers provide sufficient flow features and attack characteristics. Furthermore, 64-bits of `FlowID` in *ArOMA* provide granularity to have enough distinguished `FlowID`. However, sometimes it may create collision as it uses a hash on IP-4 tuple (source IP, destination IP, source port, destination port). It will

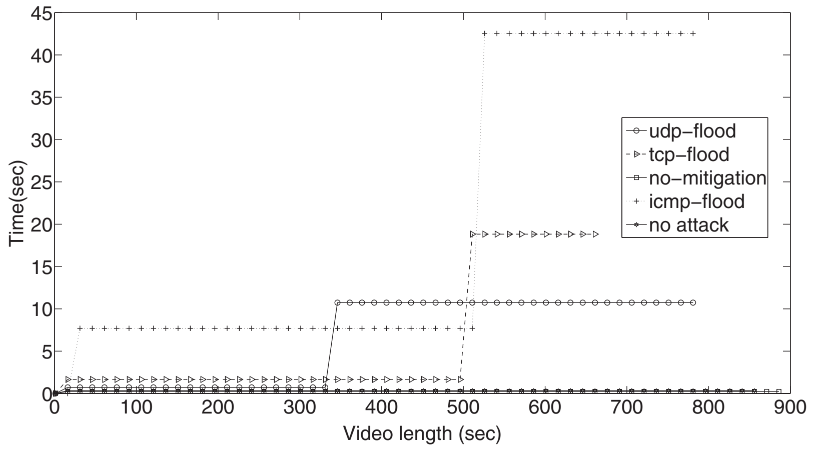

Fig. 9 – Duration of the player in paused mode.

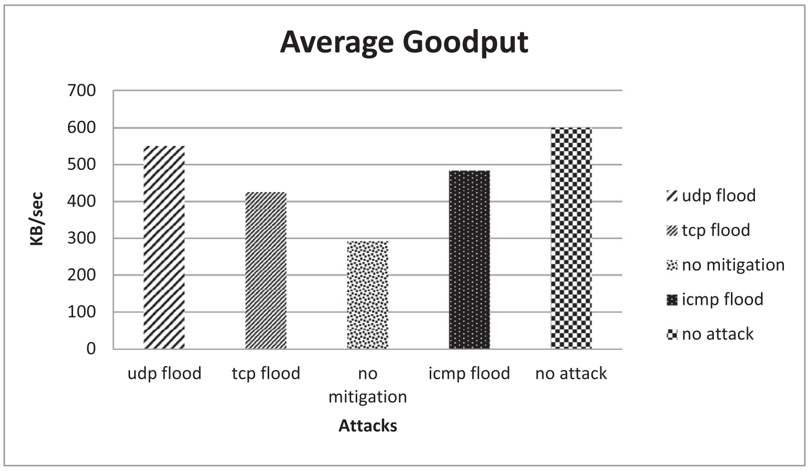

Fig. 10 – Average goodput of legitimate and attack traffic.

be the focus of our work to improve the method to avoid the collision in `FlowID` generation.

One may argue that in *ArOMA* the communication channel between ISP controller and customer controller is vulnerable to attacks. While we have to admit this fact, the current *ArOMA* assumes that customers of the ISP are behaving legitimately and they trust each other. It also ensures that every customer communicates to the ISP controller through a dedicated communication channel. In addition to pre-subscription and online authentication, other effective authentication schemes can be also deployed by the ISP, ensuring the integrity and confidentiality of alert messages sent from the customers. To cope with DDoS attacks, ISP can restrict the amount of alert messages that are sent by a particular customer in a specific time interval. All attacks targeting at SDN controllers are not considered in this work, which have already attracted lots of attention from SDN and security communities (Benton et al., 2013; Dhawan et al., 2015; Kreutz et al., 2013; Tseng et al., 2016).

Another important concern of current *ArOMA* architecture is that it is implemented with a single SDN controller in the ISP and customer network. In practice, an ISP may have a large amount of customers, making it extremely challenging or even impossible to provide real-time mitigation service by relying on a single SDN controller. Extending the current *ArOMA* to have multiple SDN controllers is an non-trivial effort, which will be the focus of our future work. For example, multiple customers may have conflicts in the installation of filtering rules. Scalability issue can be also caused by the latency resulting from controller-switch communication. To the best of our knowledge, new SDN controllers are able to handle millions of flows per second (Kreutz et al., 2015). On a commodity machine with a single CPU core, state-of-the-art controllers are capable of responding to flow setup requests within milliseconds, and Open vSwitch is capable of installing tens of thousands of flows per second with sub millisecond latency (Yeganeh et al., 2013).

8. Conclusion

DDoS attacks remain as one of the major threats disrupting today's Internet service. It is widely recognized that without effective collaborations, it is extremely difficult, if not impossible, to mitigate DDoS attacks. However, the collaboration between different domains and parties in the Internet is challenging. In this paper, we provide an on-demand DDoS mitigation framework called *ArOMA* by leveraging SDN, with an objective to facilitate the collaboration between the ISP and their customers. To demonstrate the feasibility and effectiveness of our proposed framework, we developed a proof-of-concept prototype and conducted both simulation based and testbed based experiments. The results indicated that *ArOMA* ensures that the protected asset, i.e., a video streaming service and a server, was able to maintain satisfactory performance in the presence of DDoS flooding attacks, in terms of major QoE and QoS metrics of interest.

Our experiments and analysis of the prototype have identified some key benefits of the framework, (1) the automated collaboration between customer and ISP would minimize the reaction time in an operational situation; (2) Without exposing their network topology to outside domain ISP can deploy policies to mitigate the attacks upon receiving the alert messages from its customers; (3) ISP and its customer can effectively communicate via a security API deployed at the ISP controller and exchange the alert in IDMEF format; (4) ISP is able to provide different countermeasures to the suspicious flows by redirecting them to different paths, ensuring the best QoS to legitimate flows. As such, even the suspicious traffic can finally reach the customer network, even though QoS is degraded, potentially reducing the collateral damage in the ISP network. In particular, we demonstrated that the centralized SDN controller can significantly simplify the enforcement of DDoS mitigation policies in an automated way. To improve the capability of *ArOMA*, our future work will study the scenario in which multiple customers are involved. The accurate and reliable translation from high-level security policy to OpenFlow rules will be also studied in our future research.

REFERENCES

Akbar Mehdi S, Khalid J, Ali Khayam S. Revisiting traffic anomaly detection using software defined networking. In: Sommer R, Balzarotti D, Maier G, editors. Recent advances in intrusion detection, vol. 6961. Lecture notes in computer science. Springer Berlin Heidelberg; 2011. p. 161–80.

Aljifri H, Smets M, Pons A. IP traceback using header compression. Comput Secur 2003;22(2):136–51.

Arbor Networks. Worldwide infrastructure security report. Technical report.

Bai C, Feng G, Wang G. Algebraic geometric code based IP traceback. In: 23rd IEEE international conference on performance, computing, and communications (IPCCC). 2004. p. 49–56.

Belenky A, Ansari N. On deterministic packet marking. Comput Netw 2007;51(10):2677–700.

Benton K, Camp LJ, Small C. Openflow vulnerability assessment. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking, HotSDN '13. New York, NY, USA: ACM; 2013. p. 151–2.

Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: 35th IEEE conference on local computer networks (LCN). 2010. p. 408–15.

Damianou N, Dulay N, Lupu E, Sloman M. The ponder policy specification language. In: Proceedings of the international workshop on policies for distributed systems and networks, *POLICY '01*. London, UK: Springer-Verlag; 2001. p. 18–38.

De Cicco L, Caldaralo V, Palmisano V, Mascolo S. Tapas: a tool for rapid prototyping of adaptive streaming algorithms. In: Proceedings of the 2014 workshop on design, quality and deployment of adaptive video streaming, VideoNext '14. New York, NY, USA: ACM; 2014. p. 1–6.

Demetrescu C, Italiano GF. A new approach to dynamic all pairs shortest paths. In: 35th annual ACM symposium on theory of computing (STOC). 2003. p. 159–66.

Dhawan M, Poddar R, Mahajan K, Mann V. SPHINX: detecting security attacks in software-defined networks. In: 22nd annual network and distributed system security symposium, NDSS 2015, San Diego, California, USA, February 8–11, 2014. 2015.

Dobrian F, Sekar V, Awan A, Stoica I, Joseph D, Ganjam A, et al. Understanding the impact of video quality on user engagement. Comput Commun Rev 2011a;41(4):362–73.

Dobrian F, Sekar V, Awan A, Stoica I, Joseph D, Ganjam A, et al. Understanding the impact of video quality on user engagement. In: Proceedings of the ACM SIGCOMM 2011 conference, *SIGCOMM '11*. New York, NY, USA: ACM; 2011b. p. 362–73.

Dunbar L, Zarny M, Jacquenet C, Chakrabarty S. Interface to Network Security Functions Problem Statement. Internet-Draft dunbar-i2nsf-problem-statement-01, IETF. September, 2014.

Eddy W, Clark G, Dailey J. Customer-controlled filtering using SDN. Internet-Draft draft-eddy-sdnrg-customer-filters-01, Internet Engineering Task Force, Work in Progress. February, 2016.

Fayaz SK, Tobioka Y, Sekar V, Bailey M. Bohatei: flexible and elastic ddos defense. In: 24th USENIX security symposium (USENIX security 15). USENIX Association; 2015. p. 817–32.

Feinstein B, Curry D, Debar H. The Intrusion detection message exchange format (IDMEF). RFC 4765. October 2015.

Hsiao H-C, Kim TH-J, Lee SB, Zhang X, Yoo S, Gligor V, et al. STRIDE: sanctuary trail – refuge from internet DDoS entrapment. In: Proceedings of ACM symposium on information, computer and communications security (ASIACCS). 2013.

Iimura T, Miyamoto D, Tazaki H, Kadobayashi Y. Necomatter: curating approach for sharing cyber threat information. In: Proceedings of the ninth international conference on future internet technologies. ACM; 2014. p. 19.

Ioannidis J, Bellovin SM. Implementing pushback: router-based defense against DDoS attacks. In: Proceedings of network and distributed system security symposium (NDSS). 2002.

Kijewski P, Pawliński P. Proactive detection and automated exchange of network security incidents. Abgerufen am, 20 pp, 2014.

Koutepas G, Stamatelopoulos F, Maglaris B. Distributed management architecture for cooperative detection and reaction to ddos attacks. J Netw Syst Manage 2004;12(1): 73–94.

Kreutz D, Ramos FMV, Verissimo P. Towards secure and dependable software-defined networks. In: Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking, HotSDN '13. New York, NY, USA: ACM; 2013. p. 55–60.

Kreutz D, Ramos FMV, Verssimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. Proc IEEE 2015;103(1):14–76.

Krishnan SS, Sitaraman RK. Video stream quality impacts viewer behavior: inferring causality using quasi-experimental designs. IEEE/ACM Trans Netw 2013;21(6):2001–14.

Lantz B, Heller B, McKeown N. A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM workshop on hot topics in networks, Hotnets-IX. New York, NY, USA: ACM; 2010. p. 19:1–6.

Lee SB, Kang MS, Gligor VD. Codef: collaborative defense against large-scale link-flooding attacks. In: Proceedings of the ninth ACM conference on emerging networking experiments and technologies, CoNEXT '13. New York, NY, USA: ACM; 2013. p. 417–28.

Li J, Berg S, Zhang M, Reiher P, Wei T. Drawbridge: software-defined DDoS-resistant traffic engineering. In: Proceedings of the 2014 ACM conference on SIGCOMM. New York, NY, USA: ACM; 2014a. p. 591–2.

Li J, Chang X, Ren Y, Zhang Z, Wang G. An effective path load balancing mechanism based on SDN. In: 2014 IEEE 13th international conference on trust, security and privacy in computing and communications. 2014b. p. 527–33.

Liu X, Yang X, Xia Y. NetFence: preventing internet denial of service from inside out. Comput Commun Rev 2010;41(4):255–66.

Mahimkar A, Dange J, Shmatikov V, Vin H, Zhang Y. 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. p. 24.

Mazel J, Fontugne R, Fukuda K. A taxonomy of anomalies in backbone network traffic. In: 2014 international wireless communications and mobile computing conference (IWCMC). 2014. p. 30–6 August.

Open Networking Foundation. OpenFlow switch specification version 1.4.0. ONF; 2013. Technical report.

Qin F, Tirumala A, Jon J, Gibbs K. IPERF. 2003.

Sahay R, Blanc G, Zhang Z, Debar H. Towards autonomic ddos mitigation using software defined networking. In: Proceedings of the NDSS workshop on security of emerging technologies (SENT). 2015.

Savage S, Wetherall D, Karlin A, Anderson T. Practical network support for IP traceback. Comput Commun Rev 2000;30(4):295–306.

Schnakengerg D, Holliday H, Smith R, Djahandari K, Sterne D. Cooperative intrusion traceback and response architecture (CITRA). In: Proceedings of the DARPA information survivability conference & exposition II, DISCEX. IEEE; 2001. p. 56–68.

Seufert M, Egger S, Slanina M, Zinner T, Hobfeld T, Tran-Gia P. A survey on quality of experience of http adaptive streaming. IEEE Commun Surv Tutor 2015;17(1):469–92.

Shin S, Porras PA, Yegneswaran V, Fong MW, Gu G, Tyson M. FRESCO: modular composable security services for software-defined networks. In: Proceedings of the ISOC network and distributed system security symposium (NDSS). 2013.
Snoeren AC, Partridge C, Sanchez LA, Jones CE, Tchakountio F, Kent ST, et al. Hash-based ip traceback. In: Proceedings of the 2001 conference on applications, technologies, architectures, and protocols for computer communications, SIGCOMM '01. New York, NY, USA: ACM; 2001. p. 3–14.
Spring N, Mahajan R, Wetherall D, Anderson T. Measuring isp topologies with rocketfuel. IEEE/ACM Trans Netw 2004;12(1):2–16.
Stone R. Centertrack: an IP overlay network for tracking dos floods. In: Proceedings of the 9th conference on USENIX security symposium – volume 9, SSYM'00. 2000. p. 15.
Studer A, Perrig A. The coremelt attack. In: Proceedings of the 14th European conference on research in computer security, ESORICS'09. Berlin, Heidelberg: Springer-Verlag; 2009. p. 37–52.
Tseng Y, Zhang Z, Nat-Abdesselam F. Srv: switch-based rules verification in software defined networking. In: 2016 IEEE NetSoft conference and workshops (NetSoft). 2016. p. 477–82.
Wichtlhuber M, Reinecke R, Hausheer D. An sdn-based cdn/isp collaboration architecture for managing high-volume flows. IEEE Trans Netw Serv Manage 2015;12(1):48–60.
Yaar A, Perrig A, Song D. Pi: a path identification mechanism to defend against DDoS attacks. In: 2003 symposium on security and privacy, 2003. Proceedings. 2003. p. 93–107.
Yaar A, Perrig A, Song D. SIFF: a stateless Internet flow filter to mitigate DDoS flooding attacks. In: Proceedings of the 2004 IEEE symposium on security and privacy. 2004. p. 130–43.
Yang X, Wetherall D, Anderson T. A DoS-limiting network architecture. Comput Commun Rev 2005;35(4):241–52.
Yeganeh SH, Tootoonchian A, Ganjali Y. On scalability of software-defined networking. IEEE Commun Mag 2013;51(2):136–41.

Rishikesh Sahay is PhD student at Telecom SudParis, France. He obtained his M.Tech degree from International Institute of Information Technology Bhubaneswar (IIIT Bhubaneswar) in 2011. Then he joined as a project fellow at Indian Institute of Technology Patna. His research interests include network and policy management, software-defined networking and network security.

Gregory Blanc is currently an assistant professor at Telecom SudParis in the networks department where he researches about SDN/NFV security, access control systems and alert correlation. He previously spent two years as a postdoctoral researcher at Telecom SudParis where he took the technical coordination of an EU-Japan collaborative project funded by the European Commission (FP7). He holds a Master in Network and Information Security from ESIEA, a French grande ecole and has received his PhD degree from NAIST, a Japanese national research university in 2012.

Zonghua Zhang is an associate professor (with HDR – accreditation to supervise research) of Institute Mines-Telecom of France. He is also affiliated with CNRS UMR 5157 SAMOVAR Lab. Previously, he worked as an expert researcher at National Institute of Information and Communication Technology (NICT), Tokyo. He also spent two years as a postdoctoral researcher at the University of Waterloo, Canada and INRIA, France right after obtaining his Ph.D. degree from Japan Advanced Institute of Science and Technology (JAIST). His research topics cover security management, network forensics, reputation management and security protocols, in different types of computer and communication networks, with the current scenarios about Network Function Virtualization, Software Defined Networking, and Cyber-Physical Systems. He has contributed to tens of national and international research projects, with the results published in over 60 international journals and conference proceedings. He is serving on the editorial board of Computers & Security, Security and Communication Networks, IEEE Communications Magazine, Intl. journal of Network Security.

Hervé Debar is professor at Telecom SudParis, and the Head of the Networks and Telecommunication Services Department. His activity is related to the information and communication technology (ICT) security area. He has been involved in intrusion detection research and he is currently focusing on Software Defined Networking and Network Function Virtualization, with an emphasis on automated threat mitigation.