THIAGO RODRIGUES MEIRA DE ALMEIDA

A COLLABORATIVE ARCHITECTURE AGAINST DDOS ATTACKS FOR CLOUD COMPUTING SYSTEMS

São Paulo
2019

THIAGO RODRIGUES MEIRA DE ALMEIDA

A COLLABORATIVE ARCHITECTURE AGAINST DDOS ATTACKS FOR CLOUD COMPUTING SYSTEMS

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Ciências.

São Paulo
2019

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 12 de Fevereiro de 2019.

Assinatura do autor

Assinatura do orientador

FICHA CATALOGRÁFICA

de Almeida, Thiago Rodrigues Meira

A Collaborative Architecture against DDoS attacks for Cloud Computing Systems/ T. R. M. de Almeida. – ed. rev. – São Paulo, 2019.

83 p.

Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais (PCS).

1. Software-defined networking #1. 2. Segurança de dados #2. 3. Colaboração #3. 4. Otimização #4. 5. Computação em Nuvem I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais (PCS). II. t.

AGRADECIMENTOS

Agradeço muito à minha família por todo o apoio incondicional durante essa jornada. Principalmente aos meus pais, Abelardo Almeida e Cilene Almeida, que sempre tiveram muita coragem para vencer os desafios que a vida nos impôs, nunca deixando de se preocupar com a boa formação de seus filhos; E à minha irmã, Juliana, que sempre esteve ao meu lado independentemente da ocasião. Também agradeço à mulher da minha vida, Carla Cardoso, que conheci durante a conferência onde apresentei os primeiros resultados do meu trabalho e que sempre fará parte da minha vida.

Além disso, aproveito para registrar a jornada dos meus avós maternos, José Rodrigues de Souza, que veio de Missão Velha, no sertão cearense, e Olívia Meira Rodrigues, que veio de Jequié, no interior baiano; e dos meus avós paternos, Elisa Ferreira Soares e Olmaro Laurindo de Almeida, que vieram de Ervália, interior de Minas Gerais. Se a perseverança e esperança deles por um futuro melhor para seus filhos não os tivesse trazido até São Paulo, este trabalho também não seria possível.

Um agradecimento especial ao meu orientador, Marcos Antonio Simplicio Junior, que sempre esteve a postos para me apoiar, me ensinar e me corrigir sempre que fosse necessário. A oportunidade que me deu abriu muitos horizontes com os quais eu apenas sonhava. Também presto meus agradecimentos aos amigos do Laboratório de Arquitetura e Redes de Computadores (LARC) que sempre me inspiraram com seu conhecimento.

Finalmente, sou muito grato à Ericsson Research e à Fundação para o Desenvolvimento Tecnológico da Engenharia (FDTE) por financiarem o meu trabalho. Iniciativas como essas são muito importantes para o presente e futuro da pesquisa científica em nosso país.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACL	Access Control List
CDN	Content Delivery Network
DDoS	Distributed Denial of Service
DoS	Denial of Service
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
ILP	Integer Linear Programming
IPS	Intrusion Prevention System
ISP	Internet Service Provider
JSON	JavaScript Object Notation
NAT	Network IP Address Translator
NFV	Network Function Virtualization
ODL	OpenDaylight
OVSDB	Open vSwitch Database Management Protocol
QoS	Quality of Service
RFC	Request for Comments
RPC	Remote Procedure Call
S²F	Security Service Function
SDN	Software-Defined Network

SFC	Service Function Chaining
SFF	Service Function Forwarder
SMT	Satisfiability Modulo Theory
USP	Universidade de São Paulo
XML	Extensible Markup Language

ABSTRACT

Distributed attacks, such as Distributed Denial of Service (DDoS) ones, require not only the deployment of standalone security mechanisms responsible for monitoring a limited portion of the network, but also distributed mechanisms which are able to jointly detect and mitigate the attack before the complete exhaustion of network resources. This need led to the proposal of several collaborative security mechanisms, covering different phases of the attack mitigation: from its detection to the relief of the system after the attack subsides. It is expected that such mechanisms enable the collaboration among security nodes through the distributed enforcement of security policies, either by installing security rules (*e.g.*, for packet filtering) and/or by provisioning new specialized security nodes on the network. Albeit promising, existing proposals that distribute security tasks among collaborative nodes usually do not consider an optimal allocation of computational resources. As a result, their operation may result in a poor Quality of Service for legitimate packet flows during the mitigation of a DDoS attack. Aiming to tackle this issue, this work proposes a collaborative solution against DDoS attacks with two main goals: (1) ensure an optimal use of resources already available in the attack's datapath in a proactive way, and (2) optimize the placement of security tasks among the collaborating security nodes. Regardless the characteristics of each main goal, legitimate traffic must be preserved as packet loss is reduced as much as possible.

1 INTRODUCTION

Distributed attacks (*e.g.*, Distributed Denial of Service – DDoS) are currently among the most serious network threats faced by organizations of different sizes, especially considering the amount of traffic involved and the total number of services affected worldwide. The frequency of such attacks has been steadily increasing over the last years: according to Akamai, a content delivery network and cloud service provider, when comparing the mid-year reports of 2017 and 2018, the number of DDoS attacks grew 16% and a new volumetric DDoS attack record of 1.35 Tbps in size was reached in the same period (AKAMAI, 2018). These attacks typically involve the exhaustion of available resources in the target (*e.g.*, network bandwidth or memory), a situation that is often difficult to differentiate from a simple high service usage by legitimate users. Besides such volumetric attacks, there are also threats exploiting application-layer vulnerabilities in order to turn the target unavailable (*e.g.*, SQL injection) (ANSTEE et al., 2016; AKAMAI, 2018).

Unfortunately, due to the distributed characteristic of such attacks, networks closer to its sources usually have difficulties to detect the issue because the lower traffic profiles involved do not seem to pose any threat, at least not until they are combined closer to the attack's target. This characteristic motivates the deployment of mitigation strategies that enable the cooperation among multiple security mechanisms, including traditional appliances such as intrusion detection systems (IDS), firewalls, and intrusion prevention systems (IPS). By distributing these appliances all over the network, they can gather relevant traffic information and act upon this shared knowledge. For example,

collaborative solutions may rely on different strategies for detecting, analyzing and mitigating distributed attacks, instantiate new security appliances (*e.g.* IDS, IPS, and firewalls) wherever needed, and potentially also adapt themselves to newly discovered threats aiming to better mitigated future attacks. The combination of these strategies must ensure not only the needed Quality of Service (QoS) for the protected applications, but also optimize the resource usage of the security appliances, so they do not fall victim of the attack themselves.

1.1 Motivation

Even though the specialized literature includes solutions that mitigate DDoS attacks while sharing security tasks cooperatively among their security nodes, there are few works that try to optimize the use of computational resources during the mitigation process. More precisely, the mitigation approaches employed over the malicious traffic (*e.g.*, packet dropping or rate limiting) rarely take into account aspects like (1) the best usage of computational resources already available at the nodes participating in the collaboration process, and (2) the distribution of mitigation tasks among the existing security nodes. In the end, it is common to adopt a fixed strategy, like deploying the mitigation tasks on the nodes closer to the attack's sources. Albeit useful as a first heuristic, this is not always possible (*e.g.*, because nodes closer to the source are not part of the collaborative network or may not have enough resources to handle the traffic) or even optimal in terms of resource utilization (*e.g.*, because some nodes along the attack's path may be better equipped to deal with it).

Since the usage of network resources is a critical issue in the context of DDoS and due to the fact that such volumetric attacks are being performed through novel different ways (*e.g.* IoT-related DDoS attacks (ANTONAKAKIS et al., 2017)), it must be carefully addressed by collaborative solutions for an effective attack mitigation. As an example, during a volumetric flooding attack, nodes that are already overloaded may

not be available to properly cooperate with other congested nodes, as the available bandwidth is not enough for such communication, making it difficult to distribute the mitigation tasks. Actually, in this situation the mitigation solution may aggravate network congestion, as the flow of cooperation-related messages is likely to increase resource usage on the cooperating nodes.

1.2 Goals

The main goal of this work is to specify and evaluate a security collaborative architecture against DDoS attacks. As a main characteristic, the proposed solution should take into account already available resources in a service path to ensure a DDoS attack is mitigated as earlier as possible in a given datapath. For such purpose, each collaborative element should ensure its available resources are applied in the mitigation of a DDoS attack and the controller should apply an optimal distribution of security tasks among enabled network nodes. Thus, the proposed solution deals with detection and mitigation capabilities, defining an architecture in which (1) nodes can negotiate the amount of resources they will dedicate to the security tasks, (2) new nodes can be instantiated whenever and wherever needed for better handling the attack, (3) nodes help in the collaborative process in a mode which enables the use of available resources that were not previously negotiated, and (4) security tasks are distributed by means of an optimization algorithm based on the resource availability along the datapath.

1.3 Method

This work follows the hypothetical-deductive method, making use of references available in the literature to define a research problem and then specifying a solution for addressing it. The proposed solution is then implemented as a functional prototype, so it can be evaluated and tested.

The first phase of the work consisted in the evaluation of the literature, covering collaborative solutions aimed at detecting and mitigating distributed attacks. The literature research allowed the identification of research gaps in the state-of-the-art, namely the need for optimization strategies when distributing tasks among collaborative nodes, and introduced some assumptions that guided the work. The second phase covered the identification of ways to collaboratively mitigate distributed attacks while optimizing the overall network resources usage, leading to the specification of a collaborative architecture that tackles these issues. Finally, the third phase consisted in the development of a prototype, thus allowing the proposal to be evaluated in a controlled environment.

1.4 Contributions

The main contribution of this research is the proposal of a novel solution against DDoS attacks considering the optimal allocation of resources among nodes from multiple administrative domains that compose the protected network. Such capability aims at enhancing the overall QoS in the protected network during a volumetric DDoS attack, while still preserving legitimate flows. The proposed architecture enables the deployment of different security services, such as IPS, IDS, and firewalls, focusing mainly on the consumption of computing resources in the nodes responsible for operating these security functions. The collaboration among nodes involves a negotiation capability, so they can define how much traffic they are willing to treat. This allows the establishment of different collaboration models, including not only (1) the full offload of a security task but also (2) a partial offload, hereby called a “best-effort” approach, aiming at consuming the maximum amount of already available resources in the datapath. In the latter case, whenever the amount of resources required for performing the outsourced task reaches the agreed threshold, the task is not performed by the node that engaged in the collaboration, and untreated packets are marked so they can be treated later.

1.5 Document Organization

This document is organized in chapters. Chapter 2 discusses the concepts needed to contextualize this work. Chapter 3 analyzes the state-of-the-art works related to collaborative solutions against distributed attacks, providing an overview of the addressed research problem. Chapter 4 introduces a novel collaborative solution against DDoS attacks that ensure the optimization of computational resources in one or more domains. Chapter 5 proposes different scenarios aiming at evaluating the proposed collaborative architecture. Finally, Chapter 6 presents our final considerations and ideas for future work.

2 BACKGROUND: SOFTWARE-DEFINED NETWORKING (SDN)

Software Defined Networking (SDN) (GREENE, 2009) is a technology that is used as an important basis for the present collaborative proposal and also for some similar-purpose solutions in the literature. The concept of SDN emerged as a way for providing management and control for computer networks in a centralized way through the separation of the control plane from the data plane. More precisely, differently from traditional networks, the flow tables in routers and switches, which define the actual functioning of the data plane, are managed through a piece of software installed on a trusted server called SDN Controller, allowing the logical management of the entire network using standard protocols such as Openflow (MCKEOWN et al., 2008).

One of the main benefits of the use of a SDN approach for modern data centers is the reduction of the operational costs, allowing the easy provisioning of virtualized network components. This characteristic makes SDN a highly valuable technology for cloud computing systems, since such environments require scalable and flexible network management and related network resources may be shared among a variety of independent elements (LIN et al., 2014).

2.1 SDN Communication and Management Protocols

SDN communication protocols are elements that intermediate the control and forwarding layers of an SDN architecture. Such protocols enable different elements

in an SDN architecture to communicate in a standardized manner, as well as the programmatical configuration of virtualized networks in a dynamic manner. OpenFlow is currently one of the main protocols employed for this purpose. Basically, OpenFlow identifies groups of packets, commonly known as flows, via specific header fields; these flows can then be directed through different paths, facilitating network management (MCKEOWN et al., 2008).

Even though there are SDN protocols for network elements such as routers, the OpenFlow protocol focus mainly on switches (BRAUN; MENTH, 2014). An OpenFlow-enabled switch is a forwarding device whose forwarding rules are enforced by matching network packet headers to the entries in a local flow table. By configuring such a table accordingly, network controllers can translate network management policies into the path to be taken by each flow. For example, the network controller can define that some flows must pass through specific security elements for inspection, or must be treated by load balancers. In this scenario, OpenFlow serves as the main communication protocol among those switches and the SDN Controller.

Figure 1 shows the elements involved in the communication between the data and control planes in an SDN network, using Openflow as underlying interaction protocol. In this figure, the control plane is represented by a SDN controller, which is itself composed of different applications. Those applications are responsible for enabling the programmability of the whole network. The effect of such logical processing is propagated to the data plane, represented by a SDN-enabled Switch, using Openflow.

Even though Openflow was the first SDN communication standard, other important protocols also emerged in the SDN community for covering different needs. One of the main examples is NETCONF, which is an Internet Engineering Task Force (IETF) proposal introduced by RFC 4741 (ENNS, 2006). NETCONF provides a secure way to configure and manage network elements, such as firewalls, switches or routers, via Remote Procedure Calls (RPC) while adopting XML as its main data structure pattern.

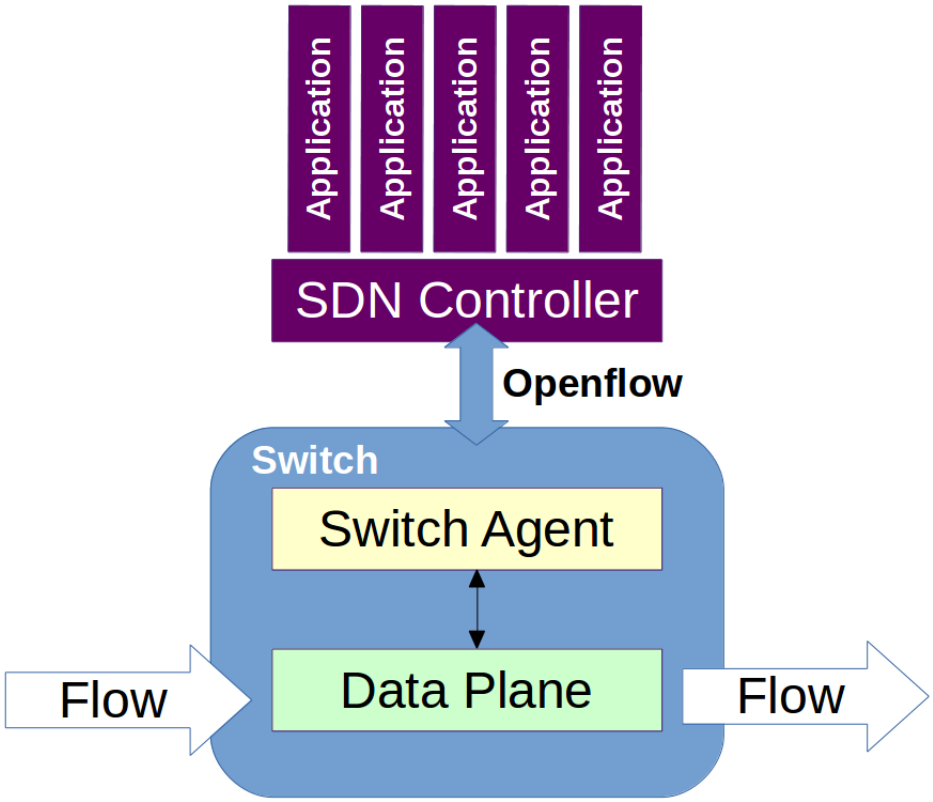

Figure 1: Openflow overview

Despite being proposed in 2006, NETCONF was adopted by the SDN community as an alternative to the traditional SNMP protocol, which is more commonly employed for managing IP networks. Indeed, it is currently a relevant protocol for configuring OpenFlow-enabled devices, following the OF-CONFIG specification (MALISHEVSKIY et al., 2014)

Besides NETCONF, OVSDB is also an open-source OpenFlow configuration protocol for managing Open vSwitch implementations; basically, it allows bridges, ports and interfaces to be created, deleted or modified on demand (PFAFF; DAVIE, 2013). OVSDB uses JSON for both its schema format and its wire protocol format. It does not handle per-flow operations, which is responsability of OpenFlow, but instead focus on datapath management.

2.2 SDN Controllers

SDN controllers are the elements responsible for managing the network's behavior, defining how data flows must be handled by SDN-enabled switches, routers and applications. Controllers usually comprise multiple modules, each of which dealing with different network functions; examples include routing, information storage, as well as security appliances able to rate limit or filter data traffic depending on its behavior. Such management and control capabilities are enabled by well-known communication protocols such as those mentioned in Subsection 2.1.

One of the first SDN controllers that appeared in the literature, NOX (GUDE et al., 2008), is an open-source platform developed alongside Openflow. NOX's architecture includes basic components for topology discovery, and is compatible with switches capable of examining incoming packets and learning the source-port mapping. More contemporary SDN controllers, with quite active development communities, include Ryu (RYU, 2014), a python-based solution, as well as Beacon (ERICKSON, 2013) and Opendaylight (MEDVED et al., 2014), both java-based platforms. The latter is of particular interest in this work, since it is quite flexible and supports advanced capabilities such as Service Function Chaining (HALPERN; PIGNATARO, 2015) — described later in Section 2.4. Basically, the Opendaylight controller is a modular open platform in which the underlying network devices and network applications are represented as objects or models. A common representation approach is to use YANG (BJORKLUND, 2010), a data modeling language for configuration and state data manipulated by the Network Configuration Protocol (NETCONF) (ENNS et al., 2011).

2.3 Data Plane and Control Plane

The main SDN paradigm is the separation of the network's data and control planes. Basically, the data plane is the set of elements that are responsible for forwarding the

traffic to the next hop along a path in the direction of the selected destination according to the control plane management. In addition to forwarding packets from one port to another, a data plane element may enforce actions on the incoming traffic, which may, for example, discard a packet due to security policies or change a packet header in consequence of a traffic control rule.

The control plane, which is represented by a logically centralized network controller in the SDN context, is responsible for defining a routing logic and managing the data plane. Such intelligence is possible due to the use of relevant information provided by the data plane elements, such as traffic monitoring statistics or packet identifiers (KREUTZ et al., 2014). This allows the definition of customizable strategies for the traffic management through the implementation of protocols such as OpenFlow (OPENFLOW, 2012) and OVSDB (PFAFF; DAVIE, 2013). It also gives the centralized manager the ability to control the behavior of the forwarding elements, defining customizable policies for implementing multiple network functions, such as firewalls, load balancers and intrusion detection/prevention systems.

Figure 2 illustrates the SDN overall architecture, showing a logically centralized controller responsible for managing the data plane elements via the Openflow protocol. Such management is generally made by enforcing packet forwarding policies in each SDN-enabled router in the network.

2.4 Service Function Chaining

A Service Function (SF) is a logical resource within a network administrative domain that is available for consumption as part of a composite service, making it possible to provide a specific treatment for received packets (HALPERN; PIGNATARO, 2015). In the context of network security, each SF usually represent a security application, such as a firewall or an Intrusion Detection/Prevention System, facilitating the provisioning

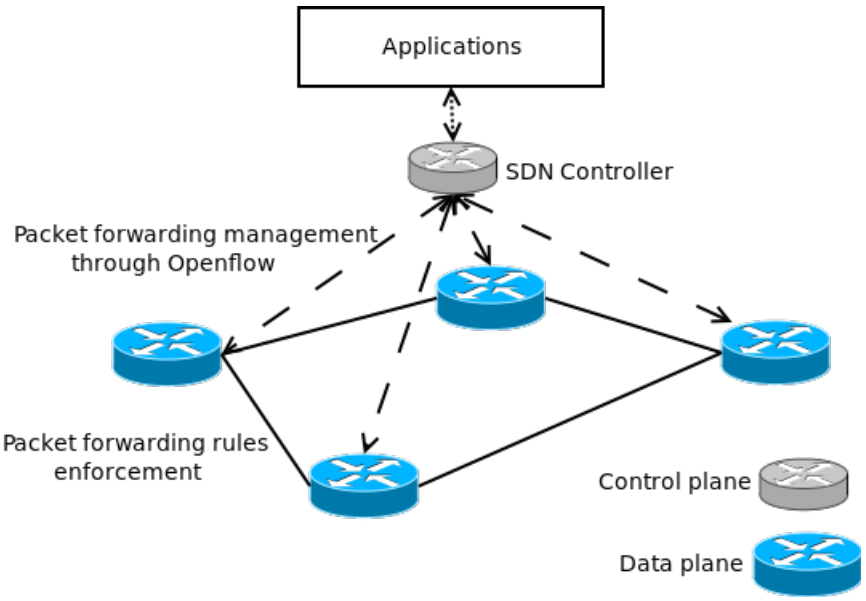

Figure 2: Overview of an SDN architecture.

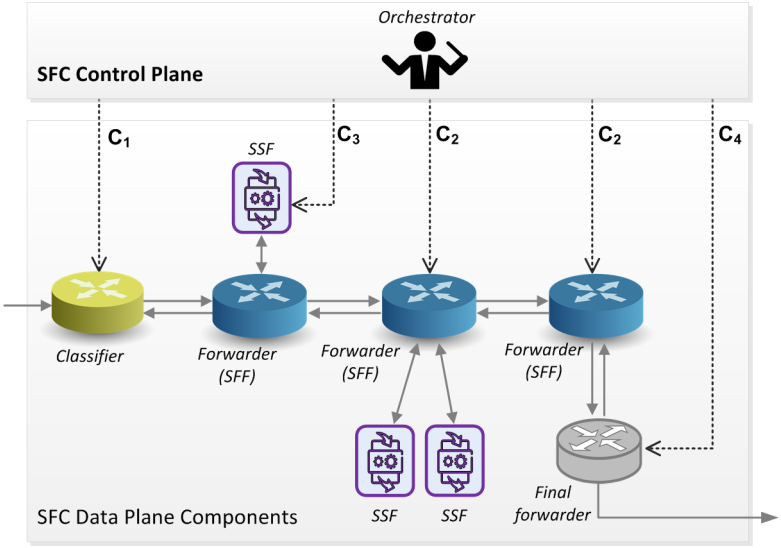

Figure 3: SFC elements overview

of security or traffic control services throughout the network. Each SF representing a security appliance is called a Security Service Function (S²F). Multiple occurrences of a given SF can exist in the same administrative domain and, since it is a logical component, an SF can be employed as a virtual element or be embedded in a physical network element (HALPERN; PIGNATARO, 2015). Figure 3 shows each element of an SFC architecture considering a security context, such as Classifiers, Forwarders (SFF) and Security Service Functions. The control plane is responsible for orchestrating those

elements located in the Data Plane through a control protocol.

The Service Function Chaining (SFC) architecture (HALPERN; PIGNATARO, 2015) was created to allow the definition and instantiation of an ordered list of SFs through which flows can be steered. It also provides several mechanisms that enable the construction and management of paths, composed by different SFs, inside a given network domain. Therefore, SFC gives network managers the ability of applying traffic control and classification rules to incoming flows, so the traffic is treated according to some predefined policies.

The main advantage of the SFC architecture is the automation of the way network connections are configured to handle flows, using different SFs. As an example, SDN controllers are able to create and manage a chain of SFs for filtering a given traffic flow depending on its source, destination or type of protocol employed. It also allows (samples of) suspicious traffic to be directed to suitable SF for analysis, such as nodes implementing Deep Packet Inspection (DPI) services. This ability is useful to implement different (even collaborative) security mechanisms for mitigating Distributed Denial of Service attacks.

2.5 SDN and Collaborative Security

The literature has several examples of how collaboration can improve security (for a survey, see (MENG et al., 2015a)). These benefits led to several initiatives based on collaboration protocols, most of which are focused on signaling the detection of DDoS attacks (MORROW; DOBBINS, 2015) or mitigating them by outsourcing tasks toward nodes closer to the attack's sources (MAHAJAN et al., 2002a; MARQUES et al., 2009). Unfortunately, however, collaboration has seen limited deployment in practice. In part, this is due to the cost of adding such capabilities in appliance routers and to the limited mitigation capabilities made available by vendors. Nevertheless, today

this issue can be addressed using SDN-based solutions (LI et al., 2014; SAHAY et al., 2015a), building upon its potential to reduce management costs and to diversify the possible interactions among different network domains. Indeed, the interest of major industry players in enabling and orchestrating collaboration among domains is behind the creation of Internet working groups focused on this precise topic, such as the Interface to Network Security Functions (I2NSF) (GROUP et al., 2017) and the DDoS Open Threat Signaling (DOTS) (MORROW; DOBBINS, 2015). Several of those initiatives are targeted at DDoS and similarly distributed attacks, which are successful when the aggregation of resources on the attackers' side overcomes the resources available at some point on the target's side (which becomes a bottleneck). As of today, a common strategy for dealing with this issue is to redirect the traffic to a scrubbing center, which is supposed to concentrate enough resources to handle the attack. With collaboration, however, the defense is also distributed, combining the capabilities of several nodes to overcome the attacker's resources and, thus, reducing the system's dependency on scrubbing centers.

This synergy between SDN and collaborative security appears in many of the collaborative security proposals in the literature. In particular, various solutions rely on a central controller with a global view of the network to create an environment ready for detecting, analyzing and mitigating attacks such as DDoS (SAHAY et al., 2015b)(LEE; KANG; GLIGOR, 2013). Such controller can be SDN-based: after all, it is usually responsible for actions that are potentially related to SDN and SFC technologies, such as coordinating the collaboration among security modules and deploying network services aiming to avoid congestion and/or to re-route flows to suitable paths. If controllers from different domains also cooperate, they can define a strategy against DDoS attacks considering alerts raised inside their own domains. For example, a controller could be able to use the SFC architecture to define a separate datapath (*e.g.*, with VLAN tags) for a suspicious flow including in this path all SFs required for

analyzing the packets (SAHAY et al., 2015b).

Despite this potential of SDN in enabling collaboration, a challenge remains in this scenario: even though collaboration is expected to benefit all entities involved, it requires one domain to allocate resources for another domain, which may not seem advantageous at first sight. To encourage such practice, each domain should be able to dynamically control the amount of resources allocated for a collaboration, negotiating it in real time and in a flexible manner. For example, a domain should be allowed to handle only a fraction of the traffic, considering its current load, to avoid undesirable situations such as resource exhaustion or hijacking, simply marking non-treated traffic that still need to be treated subsequently. Collaboration agreements established for this purpose can, however, be significantly simpler than the formal agreements usually established with scrubbing centers. After all, the collaboration may be temporary and consider only the current capabilities of each domain. Unfortunately, as further discussed in Chapter 3, existing collaboration mechanisms usually operate on an "all-or-nothing" basis, meaning that all traffic matching a given security policy must be handled by the node that accepts to perform a given task. Even though this may be enough in some cases, it is not necessarily optimal in terms of resource usage when compared to an approach that enables a partial traffic outsourcing among collaborative nodes. Thus, the solution proposed in Chapter 4 is expected to create a more flexible collaboration environment, in which resource optimization opportunities are taken into account during the establishment and maintenance of the collaboration.

3 RELATED WORKS: COLLABORATION SOLUTIONS AGAINST DDOS ATTACKS

This chapter discusses collaborative security schemes, identifying gaps that are addressed by the solution hereby proposed. Section 3.1 starts by giving an overview of collaborative security schemes, describing the phases they usually involve. Then, Section 3.2 lists some important requirements and metrics employed for comparing and evaluating such schemes. Section 3.3 then discusses different solutions available in the literature based on these metrics, identifying gaps that are addressed in our proposal.

3.1 Collaborative Security Solutions: Phases

Collaborative security mechanisms for DDoS detection and mitigation are typically built around some specific phases (see Figure 4): first, (a) they discover nodes that may participate in the collaboration; then, (b) they gather information about them to negotiate and decide a collaboration strategy; finally, (c) they deploy the required mechanisms for relieving overloaded nodes from tasks that are excessively resource-consuming. Even though existing proposals may focus on specific phases of collaboration, omitting details on how other phases are executed, these phases form the basis for the discussion of metrics that can be employed to distinguish and classify the solutions hereby surveyed.

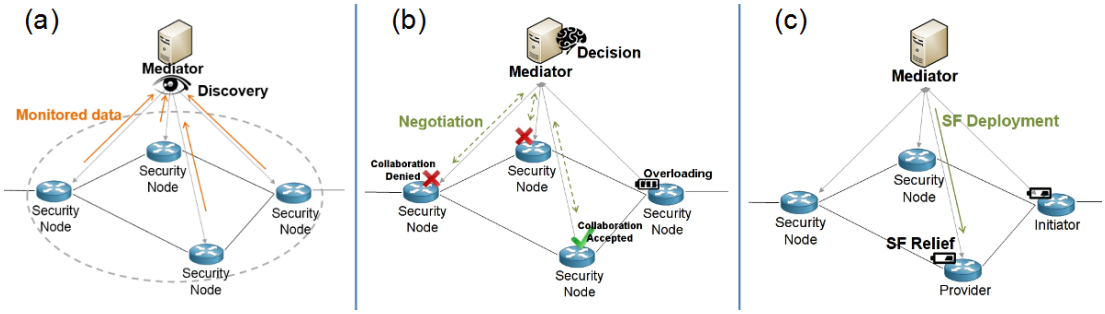

Figure 4: Collaboration phases overview: (a) Node Discovery & Information Gathering, (b) Negotiation & Decision, and (c) Deployment & Relief.

3.1.1 Node Discovery & Information Gathering

In the node discovery phase, an orchestrator or the node responsible for starting the collaboration (hereby called simply Initiator), discovers other network nodes that are capable of participating in the collaboration by sharing their available resources. For example, if the Initiator requires a firewall function, in this phase it would identify candidate nodes that are able to provide such security services using protocols such as DNS Service Discovery (DNS-SD) (CHESHIRE; KROCHMAL, 2013b). This task can be accomplished in a distributed manner, by the Initiator itself, for example by flooding some paths from/to which it is receiving/sending packets, using broadcast or multicast (*e.g.*, for DNS-SD, the Multicast DNS protocol (CHESHIRE; KROCHMAL, 2013a) could be employed). Alternatively, the Initiator could request the aid of a centralized entity, which should have a global view of the network and of the capabilities of the corresponding domain's nodes (*e.g.*, again for DNS-SD, one could employ a hybrid-proxy solution as described in (CHESHIRE, 2017)). Specially in the latter case, it is not uncommon that other parameters are also considered in the process of selecting nodes, such as resource availability (*e.g.*, CPU, memory, and/or bandwidth) (WANG et al., 2013), location in the data path (MALIALIS; KUDENKO, 2013), further details on the security functions available in those nodes (SAHAY et al., 2015b), or simply their willingness to participate in the detection/mitigation process (ZHANG; PARASHAR, 2006). Protocols commonly used to synchronize information, such as Border Gateway Protocol (BGP) (REKHTER;

LI; HARES, 2014) and Simple Network Management Protocol (SNMP) (HARRINGTON; PRESUHN; WIJNEN, 2002), may be employed for this task. In a scenario with multiple administrative domains, inter-domain discovery may also be possible, in which case orchestrators from each domain may participate in the communication.

3.1.2 Negotiation & Decision

Using the information acquired in the previous phases, the Initiator and/or orchestrator should decide which nodes will effectively participate in the collaboration and how that will be done. Quite commonly, this means that the collaborating nodes (which we call collaboration Providers) will have to run some new tasks for handling a portion of the traffic; for example, a firewall may need to install a new set of rules for filtering or redirecting packets matching certain criteria, while a router might need to deploy rate-limiting mechanisms to some of its ports. If the Provider is not already in the corresponding flows' path, the traffic needs to be redirected to it. Since those tasks and potential extra traffic may result in a higher usage of resources at those nodes, some optimization mechanism should be employed in the decision-making process. In addition, and specially when the Initiator is responsible for making the decision on which nodes will become Providers, the amount of extra load to be taken by the latter might need to be negotiated. After all, an optimal solution from the Initiator's point of view may not be considered acceptable by all Providers (*e.g.*, due to a change of their load or policies since the Information Gathering phase). This may lead to some nodes denying the collaboration request, or accepting to fulfill it only partially, which may re-trigger the negotiation and decision processes.

3.1.3 Deployment & Relief

After agreeing on a task distribution, the required security functions are installed in the designated nodes for relieving the burden at the Initiator. Technologies such

as Network Function Virtualization (NFV) (MIJUMBI et al., 2016) may be employed for handling the deployment of new tasks on those nodes, while Software-defined Networking (SDN) protocols (KIRKPATRICK, 2013) may be useful in case some traffic re-routing needs to be performed.

3.2 Metrics for Evaluating Collaborative Security Solutions

Collaborative security solutions against DDoS attacks usually differ on the specific approach adopted for accomplishing their goals. These differences include their overall architecture and interaction among components, as well as the variables considered and mechanisms employed for detecting and mitigating attacks. In what follows, we discuss in more detail those aspects, which are latter employed in Section 3.3 as metrics to evaluate and compare the state-of-the-art collaborative solutions against DDoS attacks.

3.2.1 Architecture

Collaborative security solutions, as well as its individual modules, can be organized according to different architectural models. On a high level, depending on how entities decide which and where new services should be deployed in the network, they can be classified in two main categories: centralized and distributed, described in what follows.

- *Centralized*: centralized architectures rely on a central node responsible for listening to all other security nodes and for communicating with them, issuing orders on how to proceed whenever a security incident is identified or ceases to occur. This approach usually simplify management tasks, since the central entity may have a global view of the network and be given the required administrative privileges to orchestrate its nodes, as exemplified in Figure 5. On the other hand,

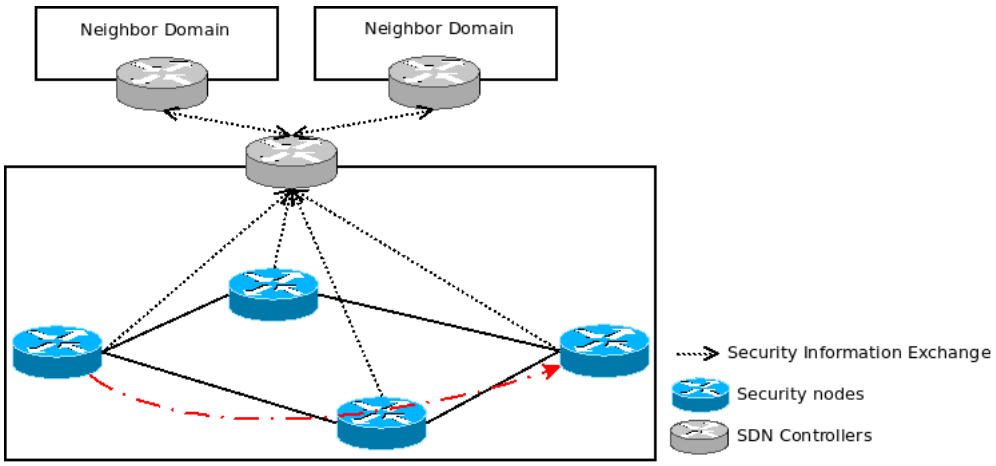

Figure 5: Centralized architecture

this also leads to potential scalability and reliability issues, as the central node may become a single point of failure and target of attacks, leading to the need of robust techniques for its protection and recovery in case of collapse. One possible approach for addressing these issues is to create a hierarchical structure among nodes (MENG et al., 2015b); in this case, even though the architecture remains essentially centralized from a collaboration perspective, the failure of controller nodes are confined to their respective regions of the network .

- *Distributed*: in distributed architectures, all enabled nodes have the same overall functions and capabilities, being organized as a peer-to-peer network. Such decentralized environment creates a more scalable system than what is usually attainable with a centralized system. However, as main disadvantages, this approach may hinder the nodes' ability of acquiring a coherent global view of the network, potentially reducing the accuracy of the detection and mitigation processes, and may also lead to performance issues due to excessive communications among peers.

We note that, even though hybrid approaches (*i.e.*, combining these two categories) are possible, we were unable to identify any such collaborative solutions in the literature.

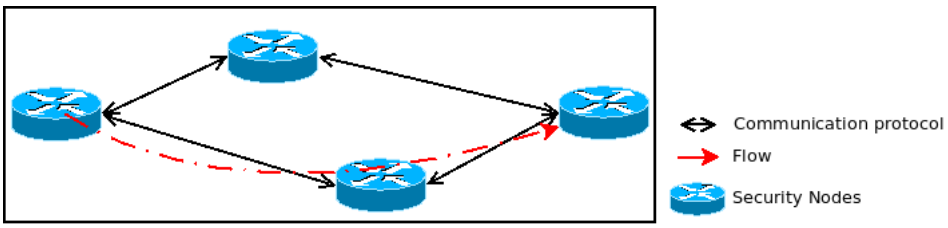

Figure 6: Distributed architecture

3.2.2 Monitored Variables

There are multiple variables that can be monitored by the system's nodes or by an external monitoring system (*e.g.*, a service orchestrator) to identify an ongoing attack and, thus, trigger a collaboration process for deploying the required mitigation services. The main ones are:

- *Bandwidth usage*: a high amount of traffic may create bottlenecks in lower-capacity network links, as well as on nodes that take longer to process packets. This may lead to a higher occupation of input and output buffers and, thus, to increased delays and packet dropping rates. The monitoring of bandwidth usage is specially important for allowing the detection of volumetric attacks (GILLMAN et al., 2015), so they can be countered before packets actually start being discarded.

- *Packet dropping rate*: indicates that a node's buffers are full, obliging it to discard packets before they are even analyzed. This metric is of particular interest for detecting DDoS attacks containing packets that, even if not too numerous, trigger resource-intensive tasks at the network's nodes (*e.g.*, deep packet inspection routines). The long time taken to evaluate the malicious traffic delays the treatment of other packets and, consequently, leads to their queuing and potential loss.

- *Protocol behavior*: an anomalous protocol behavior suggests an attack that exploits vulnerable implementations, such as those that pre-allocate resources for the treatment of packets and/or fail to release resources in spite of the attacker's anomalous conduct. A classical example refers to SYN-flood attacks against the

Transmission Control Protocol (TCP)'s three-way handshake procedure, which consists in sending several session establishment requests (SYN packets) to the target, but never actually concluding the protocol; as a result, the victim's management tables are filled with pending sessions (GILLMAN et al., 2015). The detection of protocol-related attacks usually requires a stateful security solution for keeping track of the communication status (*e.g.*, duration of a pending connection, number of connections established with the source, etc.) (GOUDA; LIU, 2005; CUPPENS et al., 2012).

3.2.3 Mitigation Capabilities

The mitigation of DDoS attacks requires suspicious traffic to be treated differently from regular traffic, preventing both the target and the nodes along the attack's path from becoming overloaded. To accomplish this goal, multiple mitigation functions can be deployed along the network. Some of them refer to to how the traffic is handled (*e.g.*, redirecting or dropping packets), while others are more related to the overall management of the network's nodes to avoid overloads (*e.g.*, offloading tasks among nodes or provisioning new ones on demand).

- *Rate limiting*: Ability to limit traffic rate when a suspicious flow is identified, based on specified security policies. Some solutions employ this approach by defining a fixed amount of rate limiting that must be imposed to a given flow (MAHAJAN et al., 2002b), while others may adopt a real-time monitoring process that allows a dynamic rate limiting according to the attack behavior (MIRKOVIĆ; PRIER; REIHER, 2002).

- *Traffic blocking*: Ability to discard the traffic that matches a given security policy. Since this is a more drastic approach than rate limiting, it is usually employed only after an attack is confirmed and the corresponding sources are properly identified.

- *Legitimate traffic tagging*: Ability to mark packets believed to be legitimate, so they can be more easily identified as such in subsequent links affected by an attack. Using such tagging mechanisms, the network becomes more capable of preserving (or even prioritizing) legitimate flows during attacks, limiting potential harms that might unwittingly come to them due to the deployment of mitigation mechanisms such as rate limiting or traffic blocking.

- *Datapath modification*: Ability to modify the traffic's path, redirecting some packets so they can be better handled by the network's security nodes. In some cases, a suspicious flow may be sent to specialized nodes aiming to deeply inspect them. In other cases, the redirection may be simply related to a node's current load, aiming to avoid congestion points.

- *Task outsourcing*: Whether or not the datapath may be modified, some collaborative solutions allow overloaded nodes to outsourcing security-related processing downward or upward the traffic flow. Often, the outsourcing of security functions is performed in the direction of the attack's source, aiming to filter malicious packets before they become too voluminous at the network's core and, thus, facilitating mitigation. Nevertheless, this capability also covers solutions in which the outsourcing is done toward the victim, which is usually done when the goal is to concentrate the analysis of the traffic on a few nodes, thus improving detection of attack patterns.

- *Node provisioning*: the collaboration process includes the ability to instantiate nodes for performing security functions deemed necessary. In some cases, the nodes are provisioned at a congestion point (*e.g.*, aiming to allow load balancing capabilities), while in other cases they can be placed in custom positions (*e.g.*, closer to the attack's origin or victim) to improve the attack's mitigation or detection efficiency. The provisioned nodes may also perform the same services as existing network nodes (*e.g.*, for enabling load balancing), or differ from those

already deployed (*e.g.*, to handle specific attack patterns).

- *Inter-domain collaboration*: some proposals explicitly take into account the possibility of enabling collaboration among neighboring domains, which may define a joint strategy and division of work to mitigate the attack. Those that do not consider this ability among its requirements sometimes may be adapted to support it, but such adaptations may not be necessarily obvious or, potentially, even feasible.

3.2.4 Collaboration Approach

Whichever the architectural model adopted, the collaboration among nodes may vary depending on which kind of information is shared among them in response to attacks. Two main approaches appear in the literature:

- *Reactive*: the Initiator sends requests of new security services (*e.g.*, traffic filtering or packet inspection rules) that need to be deployed in the network. The nodes receiving the requests (*i.e.*, Providers) may then decide to fulfill or deny them depending on its currently available resources or capabilities (see Figure 7). Nodes may also deny requests due to conflicts with other services already been performed (*e.g.*, a bandwidth limitation service may go against a packet priority policy). Hence, even though this approach is potentially effective to quickly alleviate the Initiator's burden during attacks, it requires some management effort from the Provider for dealing with conflicts. In addition, unless well orchestrated, it may also lead to inefficiencies, with multiple redundant security services being deployed along the network. For this reason, whereas some collaborative solutions do allow nodes to communicate directly, several prefer to perform this communication through a logically centralized orchestrator that can aid with conflict management.

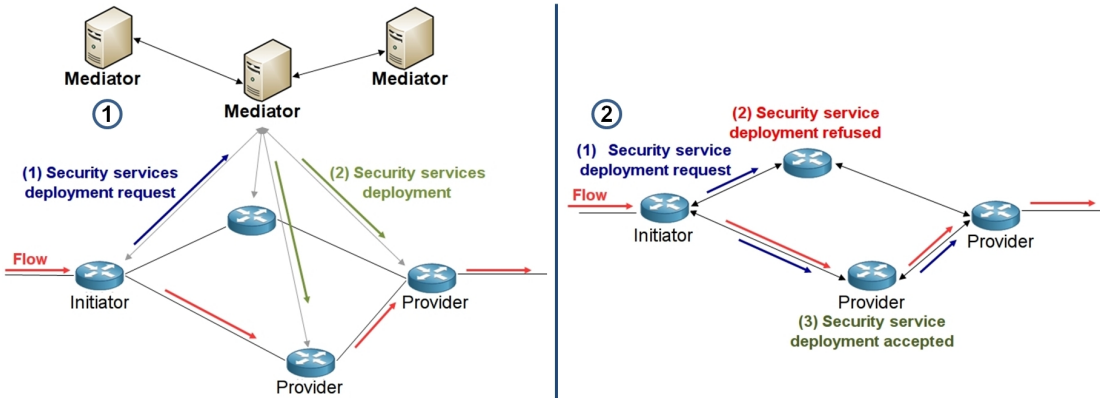

Figure 7: Reactive collaboration overview: (1) centralized and (2) distributed approaches.

- *Proactive*: instead of exchanging explicit requests for new security services, the collaboration consists in the exchange of network information, such as security alerts or a node's current load. Based on such external information and also in local traffic analysis, the network nodes and/or orchestrator may then decide to provide some mitigation service (see Figure 8). This decision may be done in a completely autonomous manner or require some kind of acknowledgment. For example, it may need the authorization from the system's orchestrator in a centralized architecture, or from other network nodes that will be somehow affected by the security service deployment in a distributed architecture. Contrasting with the reactive approach, in which the Provider is not necessarily aware of the reason behind a request, the proactive approach gives nodes more freedom to decide which would be a suitable mitigation strategy that does not conflict with its local services. It is also potentially more adequate for implementing collaboration in multiple administrative domains, as nodes from one domain (including orchestrators) are unlikely to be allowed to directly request the installation of services in nodes from another domain. Nevertheless, orchestration is still necessary to reduce inefficiencies and also to ensure that the mitigation service employed by the Provider is enough to reduce the burden of the network.

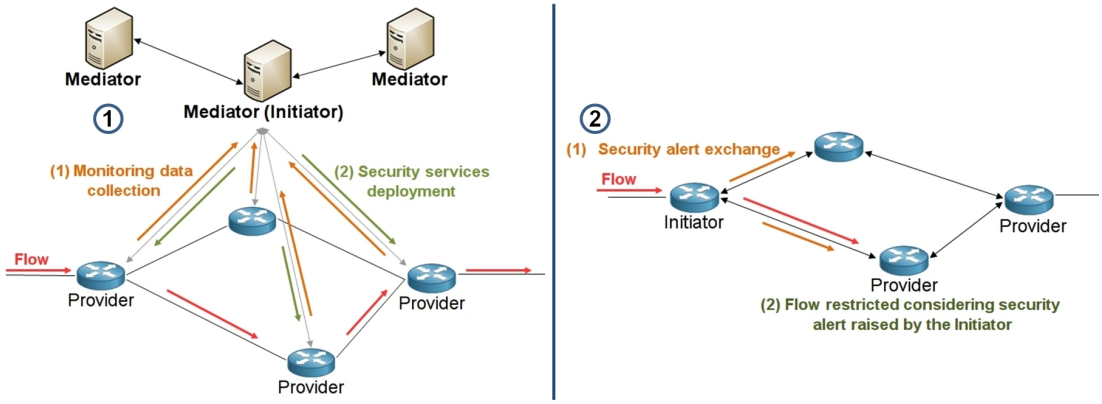

Figure 8: Proactive collaboration overview: (1) centralized and (2) distributed approaches.

3.3 Survey of Solutions in the Literature

Collaborative security solutions against DDoS attacks usually differ on the specific approach adopted for accomplishing their goals. These differences include their overall architecture and interaction among components, as well as the variables considered and mechanisms employed for detecting and mitigating attacks. In what follows, we discuss in more detail those aspects, which are employed as metrics to evaluate and compare the collaborative solutions hereby surveyed.

3.3.1 Centralized Schemes

Centralized collaborative approaches have a central entity responsible for the decision phase and deployment of security services in all enabled nodes that compose the system. This centralized controller node may have complete or partial control over the network: in the first case, it uses a global monitoring system to detect attacks, and then proactively requests the deployment of new services for mitigating them; in the second case, the central node acts simply as an intermediate, forwarding the requests made by other nodes in a reactive manner. The monitoring elements themselves can be centralized (*e.g.*, placed together with the controller node, as in (MIRKOVIĆ; PRIER; REIHER, 2002)), or distributed (*e.g.*, alongside each security node, as in (MARQUES

et al., 2009)). Whichever the case, the central node is the entity responsible for the Discovery and Information Gathering phase of the collaboration, keeping a general view of the network's infrastructure and of its nodes' status. Consequently, it is better placed to devise a suitable strategy to efficiently mitigate distributed attacks, as well as to potentially avoid redundant monitoring and communication overheads.

3.3.1.1 Overview

In a nutshell, the centralized solutions hereby evaluated can be described as follows:

- D-WARD (MIRKOVIĆ; PRIER; REIHER, 2002): a security solution deployed alongside the routers that serve entry points of different, independent network domains. The D-WARD nodes then monitor flows passing through the corresponding routers and coming from a set of pre-configured addresses. The two-way traffic involving such addresses is then compared with flow models considered normal, aiming to identify whether the network is the source of attacks. Whenever an attack is detected, rate limiting mechanisms are deployed for the suspicious flows, reducing DDoS traffic toward the rest of the Internet. Even though the collaboration is limited to D-WARD nodes and the routers they monitor, without any communication among D-WARD nodes themselves, this can be seen as a preliminary collaborative solution where each D-WARD assumes the role of the central monitoring entity in its own administrative domain.

- DiCoDefense (ZARGAR; JOSHI, 2010; ZARGAR; JOSHI, 2013): a centralized solution designed to detect DDoS flooding attacks. In each domain protected by DiCoDefense, a central node is responsible for analyzing information gathered from multiple monitoring elements deployed in the network, and then provide an adequate response based on its security policies. The solution can be deployed in multiple domains, allowing them to collaboratively identify the origin of attacks and, thus, mitigate them closer to their sources.

- NetFuse (WANG et al., 2013): a centralized mechanism for preventing traffic overloads. Even though it was designed specifically for OpenFlow-based data center networks, it can also be useful in more general SDN-based environments. NetFuse's central node acts as a proxy between OpenFlow switches and their SDN controller. It then passively listens to control messages and, potentially also queries the switches for network resource utilization (interface load and queue size), as well as fine-grained flow information. When an overloading situation is identified from the acquired data, NetFuse issues flow control rules to switches, so they can apply rate limit mechanisms or redirect suspicious flows to specialized boxes for treating packets accordingly.

- BGP Flowspec (MARQUES et al., 2009): automates inter-domain coordination of traffic filtering rules aiming to avoid attacks, such as DDoS. This information is conveyed via the BGP protocol, reusing the corresponding administrative processes and inter-domain agreements. Each BGP-enabled router uses the exchanged information to deploy rate-limiting, traffic blocking or traffic redirection rules to suspicious inbound flows.

- MF (SAHAY et al., 2015b): designed for software-defined networks, MF consists in a collaborative mitigation framework for allowing an administrative domain that detects anomalous traffic to request mitigation services from another domain. It involves the deployment of a detection engine in each domain's SDN controller, enabling the analysis of flow statistics information collected from SDN switches. Whenever an attack pattern is identified, the SDN controller not only deploys suitable mitigation policies in the domain's switches, but also shares relevant information with other affected domains. Those domains can then label the suspicious flows and redirect them to middleboxes specialized on the treatment of the corresponding threats.

- ColShield (JINGLE; RAJSINGH, 2014): a detection and mitigation mechanism

against DDoS flooding attacks in wireless mesh networks. It relies on a logically centralized architecture, with a group of local Intrusion Prevention and Detection Systems (IPDS) deployed near the network's local routers and a global IPDS installed close to the gateway router. The global IPDS, besides keeping a global view of the routers and of the local IPDS, is responsible for node admission in the network, allocating an initial bandwidth profile for each of them. For scalability reasons, the traffic monitoring tasks are performed by the local IPDS. When an attack is detected, the local IPDS first block the malicious flow on the region under their responsibility, and then inform the global IPDS of the attack; the latter then blocks future traffic involving the attacker and proceed with the corresponding nodes' bandwidth revocation.

- NetSecu (CHEN; MU; CHEN, 2011): forms a collaborative architecture of traffic control elements implemented at the network's switches, dynamically enabling and configuring security elements for attack mitigation. In a multi-site deployment, NetSecu nodes are organized in a tree-like manner, with the lower levels detecting security events and informing the higher levels until the notifications reach a central management site (CMS) at tree's root. The CMS is then responsible for defining a suitable mitigation strategy and for applying the corresponding security policies downward the tree.

- CoDef (LEE; KANG; GLIGOR, 2013): a link-flooding defense mechanism that enables routers to distinguish legitimate from attack flows, even if the latter have a low packet rate. CoDef relies on multi-path routing, responding to requests from congested nodes by redirecting traffic to underloaded links. As a result, legitimate traffic is preserved; meanwhile, low-rate malicious traffic is either (1) rerouted from the congested link, alleviating the attack, or (2) does not comply with the re-routing request, and is thus discovered and bandwidth-limited. In each CoDef-enabled administrative domain, a central route controller works side-by-

Table 1: Analysis of centralized security collaborative solutions.

	Metrics	Solutions							
		Centralized							
		Proactive					Reactive		
		D-ward	Flowsp.	Dicodef.	Netfuse	MF	Netsecu	Codef	Colsh.
		(MIRKOVIĆ; PRIER; REIHER, 2002)	(MARQUES et al., 2009)	(ZARGAR; JOSHI, 2013)	(WANG et al., 2013)	(SAHAY et al., 2015b)	(CHEN; MU; CHEN, 2011)	(LEE; KANG; GLIGOR, 2013)	(JINGLE; RAJSINGH, 2014)
Monitored Variables	Packet drop. rate	✓	✓	✓	✓	✓	✓	✓	
	Bandwidth usage		✓	✓	✓	✓	✓	✓	✓
	Protoc. behavior	✓	✓		✓		✓		
Management Objects	Rate limiting	✓	✓	✓	✓	✓	✓	✓	✓
	Traf. blocking		✓	✓		✓	✓	✓	✓
	Legit. traffic tag.	✓				✓		✓	
	Task outsourcing		✓		✓	✓		✓	
	Datapath modif.		✓		✓	✓		✓	
	Node provis.				✓	✓			
	Inter-dom. coll.		✓	✓		✓	✓	✓	

side with existing routing protocols, complementing it with re-routing and rate control mechanisms. Route controllers from neighbor domains exchange route- and rate-control requests, identifying routers that can handle those messages accordingly.

3.3.1.2 Analysis

As can be seen in Table 1, centralized solutions with a proactive collaboration approach are more common than reactive ones. This strategy is probably motivated by the fact that the central coordination node can take advantage of its global view of the network to take more satisfactory decisions. Hence, the other nodes can focus on monitoring and mitigation tasks rather than taking decisions on which flows need to

receive especial treatment.

The analysis of the schemes shows that they usually differ in terms of variables monitored and employed in the identification of DDoS attack patterns. Among those variables, those that appears most commonly are packet dropping rate and bandwidth usage; indeed, both of them are taken into account by all solutions except D-WARD (MIRKOVIĆ; PRIER; REIHER, 2002) and ColShield (JINGLE; RAJSINGH, 2014), which take into account only one. This is somewhat comprehensible, since a large packet dropping rate potentially has a considerable impact on the network's ability to provide its services, and it is also a strong indicator of the existence of overloaded nodes (even without a global network view); meanwhile, a high bandwidth usage is good indicator of volumetric DDoS attacks, which are quite common worldwide (AKAMAI, 2017), and usually a cause of an increased packet drop rate. Possibly for this reason, even though one of the surveyed solutions relies on packet dropping rate alone (namely, ColShield (JINGLE; RAJSINGH, 2014)), the majority of them uses a combination of packet dropping rate, bandwidth usage and/or transport protocols anomalies. Nevertheless, among the centralized solutions analyzed, only NetFuse (WANG et al., 2013), BGP Flowspec (MARQUES et al., 2009), and NetSecu (CHEN; MU; CHEN, 2011) take into account all three variables when detecting DDoS attacks.

In terms of mitigation capabilities, rate limiting is by far the approach most commonly adopted in centralized collaboration proposals, being present in all solutions evaluated, followed by traffic blocking, which appears in 6 out of 8 solutions. The preference for this mitigation strategy is justified by the fact that limiting the rate of suspicious flows may be enough to lessen the effects of DDoS attacks without too much impact on flows that might later be identified as legitimate. This is, thus, a conservative approach. When a flow is confirmed as malicious, on the other hand, the more extreme measure of blocking it completely can then be applied.

The desire to preserve legitimate traffic is also the motivation behind solutions such

as D-WARD (MIRKOVIĆ; PRIER; REIHER, 2002), MF (SAHAY et al., 2015b) and CoDef (LEE; KANG; GLIGOR, 2013) to allow legitimate flows to be tagged as such, so they can receive preferential treatment. In D-WARD (MIRKOVIĆ; PRIER; REIHER, 2002), for example, packet classifier components are deployed all along the network for allowing the differentiation between good and malicious packets; this avoids the need of repeated verifications on legitimate flows by multiple nodes, so the corresponding packets can be handled more efficiently after being tagged. Interestingly, this approach can also be seen as extreme if we consider that flows considered legitimate may be later discovered to be malicious. Therefore, the strategy should be considered with care, similarly to the more commonly adopted traffic blocking mechanism.

Leveraging the global view of the network, typical of centralized collaborative solutions, some proposals in this category also include more advanced mitigation mechanisms related to task outsourcing. For example, datapath modification is usually employed to avoid nodes from getting their resources exhausted: when (or, preferably, before) that happens, part of the traffic being handled by the overloaded node is redirected to another node (MARQUES et al., 2009). The node that assumes the burden of the mitigation task (*e.g.*, traffic filtering) may be any node with available resources and capable of performing the corresponding task; in some cases, the node is actually specialized in treating the identified traffic pattern (WANG et al., 2013). Additionally, a few proposals (namely, NetFuse (WANG et al., 2013) and MF (SAHAY et al., 2015b), both categorized as proactive) include the flexibility of instantiating new nodes on demand whenever deemed necessary, resulting in a more scalable solution. Those nodes could, for example, be provisioned closer to the datapath's starting points (*i.e.*, closer to attackers), allowing suspicious traffic to be treated earlier and, hence, saving network resources more effectively. We note, however, that 4 out of the 8 centralized solutions analyzed actually do not support any outsourcing mechanism, relying simply on the application of new services to mitigate attacks rather than trying to distribute

existing tasks more efficiently. Also, none surveyed solution used the overall network view provided by a centralized entity to perform an analysis considering the individual available resources to better distribute the traffic among collaborative nodes, which is an important functionality to optimize the resource usage. Such ability would enable a partial outsourcing strategy that would make it possible to take advantage of already existing unused resources distributed in a datapath, helping in the mitigation process and making it unnecessary to instantiate new nodes during an attack until all existing resources are exhausted.

Finally, only a few centralized solutions available in the literature have been developed considering a single administrative domain. The majority of them, however, explicitly consider mechanisms that allow the collaboration to span multiple domains, so they are well adapted for usage in federated scenarios.

3.3.2 Distributed Schemes

In distributed collaborative solutions, the "Decision & Deployment" phases are performed by the nodes without the help of a central authority, so the nodes themselves need to decide the best mitigation strategy. This can be done: (1) in a reactive manner, with the Initiator node defining a mitigation strategy that satisfies its needs and outsourcing it to neighboring nodes, so the former does not need to apply them itself; or (2) in a proactive manner, with nodes independently deciding to offer mitigation mechanisms based on its local information and on security alerts received from other nodes. Since these solutions lack an element with a global view of the network and higher administrative privileges, collaboration in this case is usually restricted to immediate neighbors. Even though collaborative decisions can be incrementally spread across the whole network in a hop-by-hop manner, the lack of coordination makes it potentially more difficult to achieve optimal mitigation strategies and restricts the type of actions that may be taken. These constraints, discussed further in what follows, are

possible reasons for the lower number of distributed solutions in the literature when compared with centralized ones.

3.3.2.1 Overview

The following distributed schemes are analyzed in this section:

- Pushback (MAHAJAN et al., 2002b): a distributed architecture for mitigating volumetric DDoS attacks. Pushback allows congested end hosts to collaborate with the network routers from which the excessive traffic originates, requesting the nodes in the attack's path to rate-limit traffic addressed to that host. Since routers cannot unequivocally determine if a packet actually belongs to a legitimate or malicious flow, Pushback relies on the concept of "Aggregate-based Congestion Control": an aggregate is defined as a subset of traffic with an identifiable property related to the congestion, and Pushback-enabled routers are responsible for rate-limiting those aggregates.

- Cossack (PAPADOPOULOS et al., 2003): a distributed approach whose components are deployed both at the edge of the network and at its core. The former are responsible for monitoring both ingress and egress traffic, while the latter receive and evaluate the monitored information aiming to identify the attack's origin and to define a suitable mitigation strategy. Whenever an attack is identified, filters are deployed at the attack's origin and victim networks for rate-limiting and/or blocking malicious flows.

- DefCOM (OIKONOMOU et al., 2006): forms a peer-to-peer network of defense nodes over the Internet core and through edge networks. The goal of these nodes is to preserve quality of service for legitimate traffic while reducing the occurrence of false positives when detecting attack events. The solution is composed by: alert generator nodes, placed at the victim's side, which are responsible for

detecting DDoS attacks, deliver the attack signature to the rest of the network, and sending rate-limit requests toward the attack's sources; core nodes, responsible for rate-limiting any suspicious traffic that matches the signature specified by the alert generator nodes; and classifier nodes, which ensure preferential service for legitimate traffic, by dedicating more bandwidth to them.

- DCMA (ZHANG; PARASHAR, 2006): consists in an overlay network whose nodes collect useful information related to DDoS attacks. The information is shared among nodes using a gossip protocol, so the aggregated information obtained by each of them gives an overall view ongoing traffic anomalies, improving the accuracy of attack detection routines. Whenever an attack is detected, rate limiting mechanisms are deployed on the intermediate network region between the victim and the origin of the malicious traffic.

- DDF (YOU; ZULKERNINE; HAQUE, 2008): a defense system deployed at edge routers of different administrative network domains comprising three main modules: a DDoS detection component, which identifies high bandwidth traffic patterns at the victim's end; a traceback component, responsible for identifying the edge routers at the attack's source domain; and a traffic control component, which requests (at the victim's end) and handles (at the attacker's end) the deployment of rate limiting mechanisms at the attack sources' edge routers. During the attack, the traffic control components collaborate directly and continuously, intensifying the rate limiting for more aggressive flows and alleviating such restrictions after the attack subsides.

- Firecol (FRANÇOIS; AIB; BOUTABA, 2012): creates a ring-like, overlay protection network of IPS around registered customers, protecting them against flooding attacks. Each ring consists of IPS that are at the same distance (in number of hops) from the potential victim, measuring its threat level based on the overall bandwidth supported by it versus the amount of traffic being directed to it. The

IPSs collaborate by exchanging belief scores on potential attacks; when the danger is considered high, the ISPs in the ring communicate directly to confirm whether the actual amount of traffic being directed to the target is above the latter's capacity. Attack mitigation is then performed by blocking flows coming from suspected sources, applying filters as close as possible to them.

- CRL (MALIALIS; KUDENKO, 2013): follows a decentralized approach that allows a node that detected a DDoS attack to deploy rate-limiting functions to its neighbors. if the defined limits are not enough to mitigate the attack, it can be gradually reinforced to reduce the corresponding traffic data rate.

- Poseidon (COMPAGNO et al., 2013): activates a pushback strategy once a detection threshold is reached, allowing the propagation of mitigation alerts as close as possible to the attack source. The detection mechanism considers as metric mainly packet loss during a given period of time. The mitigation strategy then imposes a rate limit over suspicious traffic until the attack is not detected anymore.

3.3.2.2 Analysis

A first observation is that, unlike centralized collaborative solutions, distributed schemes are more commonly reactive than proactive, as shown in Table 2. This approach is probably motivated by the fact that attack detection in these schemes is usually done by nodes close to the victim, which are also well placed to determine a reasonable mitigation strategy (*e.g.*, how much a given flow needs to be rate-limited considering the victim's capacity). Therefore, it is fair to let that node request services from their counterparts rather than simply send alerts to them. Nevertheless, discretion is necessary from the Provider nodes to actually verify if the request is legitimate and reasonable, since otherwise a dishonest node may cause havoc in the network (*e.g.*, filter packets that are not causing any damage or even directed to hosts under its responsibility, simply aiming to prioritize its own domain's traffic). In comparison,

Table 2: Analysis of distributed security collaborative solutions

	Metrics	Solutions							
		Distributed							
		Proactive		Reactive					
		Cossack	DCMA	Pushback	DefCOM	Firecol	CRL	DDF	Poseidon
		(PAPADOPOULOS et al., 2003)	(ZHANG; PARASHAR, 2006)	(MAHAJAN et al., 2002b)	(MIRKOVIC; ROBINSON; REIHER, 2003)	(FRANÇOIS; AIB; BOUTABA, 2012)	(MALIALIS; KUDENKO, 2013)	(YOU; ZULKERNINE; HAQUE, 2008)	(COMPAGNO et al., 2013)
Monitored Variables	Packet drop. rate	✓	✓	✓	✓	✓	✓	✓	✓
	Bandwith usage		✓			✓	✓		
	Protocol behavior	✓				✓			
Mitigation Capabilities	Rate limiting	✓	✓	✓	✓	✓	✓	✓	✓
	Traffic blocking	✓				✓			
	Legit. traf. tagging			✓	✓			✓	
	Task outsourcing			✓	✓	✓	✓		✓
	Datapath modif.								
	Node provisioning								
	Inter-dom. collab.	✓	✓		✓		✓	✓	

proactive solutions take their decisions based on the collection of alerts raised by the network, so such issues are potentially easier to handle using consensus algorithms.

The variables usually monitored in distributed solutions are similar to those found in centralized proposals, with packet dropping rate being by far the most common, while bandwidth usage and protocol anomalies appear as secondary items. The mitigation actions taken, on the other hand, have some noticeable differences. Specifically, even though the conservative approach of rate limiting continues to be the most common, the more radical strategies of blocking traffic and tagging packets considered to be legitimate are proportionally quite less frequent. The probable reason is that the latter can be more easily abused without a central authority coordinating their deployment, so their adoption in distributed environments can be considered more risky. In addi-

tion, whereas the task outsourcing capability is quite common among the distributed solutions surveyed, appearing in all reactive proposals, none of them includes datapath modification and node provisioning features. This gap is probably due to the fact that providing such services efficiently usually requires some nodes with administrative privileges over the network, as well as a broad view of resources usage by existing nodes, both not commonly found in highly distributed environments.

Similarly to centralized solutions, support for multi-domain collaboration is also frequently present in distributed schemes, making this latter approach also suitable for federated networks. However, the lower degree of coordination typical of distributed solutions usually limit the inter-domain collaboration to nodes at the domains' frontiers, although the decisions thereby taken can be later propagated further inside the domain by its own nodes.

4 PROPOSED SOLUTION

This chapter presents the Collaborative Solution against Distributed Denial of Service Attacks that constitutes the main contribution of this work. The proposed architecture focus mainly on volumetric infrastructure-layer DDoS attacks, which, according to Akamai (AKAMAI, 2017), represented 99.41% of the attacks registered in 2017 by leading Content Delivery Networks (CDN) and Cloud Service Providers.

We start the discussion in Section 4.1 by presenting a motivational scenario, which can be seen as a possible target of the proposed solution. Then, Section 4.2 presents the functional and nonfunctional requirements that can be drawn from this scenario. Finally, Section 4.3 describes the proposed solution and describes its main characteristics.

4.1 Scenario

This section describes and discusses a DDoS motivational scenario to contextualize the proposed collaborative solution. It consists in the implementation of a web server in a cloud infrastructure aiming at leveraging its elasticity to mitigate DDoS attacks, using the Service Function Chaining (SFC) architecture described in Section 2.4 to instantiate useful security-related service functions. The example, illustrated in Figure 9, considers a web server protected by a chain of four Security Service Functions (S²F): *S²F4*, a Web Application Firewall (WAF) hosted on the web server; *S²F3*, a gateway firewall at virtual data center (VDC)'s entry point; *S²F2*, a firewall placed at the cloud's entry point; and several instances of *S²F1*, which are firewalls in the domain of the

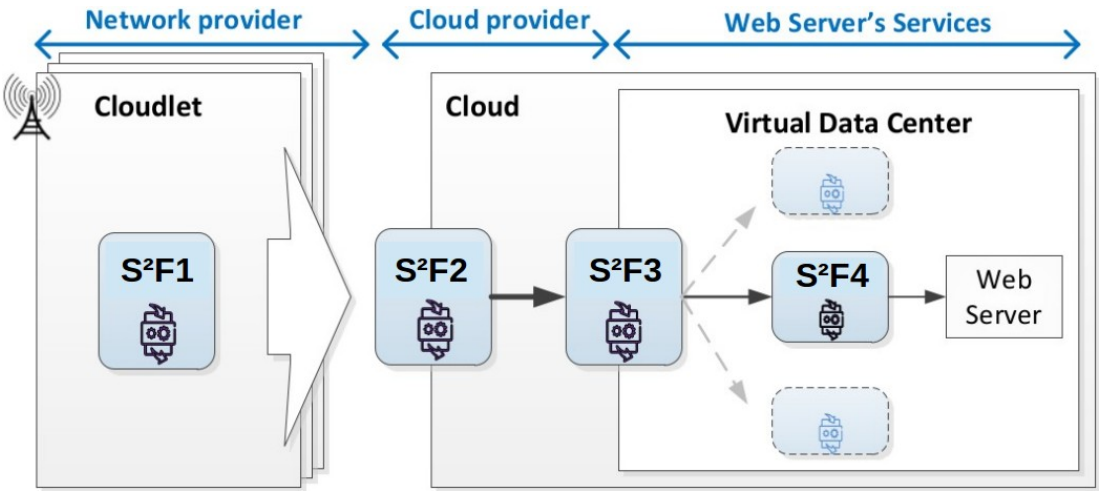

Figure 9: Collaboration Scenario

Internet Service Providers (ISPs), called as Cloudlet in the presented scenario, and, thus, close to the web server's clients. Although in practice multiple cloudlets are likely to exist to filter malicious traffic before reaching the cloud, for simplicity we simply refer to *S²F1* as one single S²F along our example. The goals of collaboration in this scenario are: (I) mitigate DDoS attacks as close as possible to the source, so their overall impact (*e.g.*, in terms of latency and drop rate for legitimate packets) remain minimal; (II) reduce the infrastructure costs by avoiding the excessive computational resources consumption caused by packets that will be dropped later, so each S²F can be dimensioned accordingly; and (III) make detection and mitigation tasks dynamic and fast by enabling different domains to share resources for such activities (*e.g.*, information provided by the WAF may trigger mitigation tasks on the cloudlet).

To work with a more concrete example, suppose for simplicity that all firewalls are dimensioned at 100σ, where σ is a generic measure of computational resources. Suppose also that, at some point in time, their resource usage for processing their own security tasks is 50σ. Then, due to a DDoS attack, the resource usage at S²F1 and S²F4 rise both to 80σ, for example; for the same reason, if S²F2 and S²F3 keep processing the same tasks, their resource usage would become respectively 55σ and 105σ, as S²F3 would end up filtering most of the DDoS traffic as currently configured. In this case,

the network packet queues in S²F3 would continuously grow, increasing latency, until its buffers are full and packets begin to be dropped. One could deal with this issue by creating a second instance of S²F3 and attaching both instances to a load balancer; it would be better, however, if S²F3 could simply *offload* some tasks (taking 25σ, for example) to S²F2, so both would end up with a resource usage of 80σ, keeping the QoS without any extra instance provisioning. With a multi-domain collaborative solution as hereby proposed, that is exactly what would be done to deal with the attack.

Some time later, when the DDoS is already under control due to the collaboration, the attack pattern changes and starts including packets that are only filtered by rules installed at the Security Service Function S²F4. This raises the resource usage of the corresponding filtering task τ from 0σ to 30σ (and, thus, S²F4's resource usage becomes 110σ. Unfortunately, S²F4 cannot offload the whole task to any other firewall, as that would just change the place of the problem. Nevertheless, it can collaborate with some of them for sharing the corresponding load by combining a "best effort" approach with optimization oportunities. For example, if τ would take the same amount of resources wherever executed, S²F4 could ask S²F3 and S²F2 to invest 10σ each on this task, so the load on S²F4, S²F3 and S²F2 would be 90σ. It is possible, however, that running τ with only 10σ at S²F2 will not be as effective as in S²F4, since the volume of traffic at S²F2 is higher; after all, in S²F4, τ runs only on packets that have not been filtered by S²F3's tasks, whereas S²F2 sees the full load of packets to which τ must be applied even if S²F2's regular filtering tasks are all applied prior to τ. Or even better, S²F4 could raise the need for collaboration for a centralized controller that is able to verify which are the best nodes that are able to collaborate based on their current resource usage through a resource optimization approach. Therefore, it becomes possible to decide a reasonable workload division that ensures no single firewall gets overloaded.

Finally, there may be situations in which neither offloading or load-sharing is fea-

sible, so a new S²F needs to be instantiated to avoid upgrading the existing machines (vertical scaling). Even in this case, a multi-domain collaborative approach provides more flexibility than simply using load balancing, since the new instance added to the chain does not need to be an exact copy of some existing S²F. As a result, some optimizations are possible. For example, the new S²F could be placed at the beginning of the chain and receive tasks that can filter many packets (preferably with little processing), thus reducing the network's overall load.

4.2 Requirements

From the scenario described in Section 4.1, we can draw the following functional requirements for the proposed solution's design:

1. Offload capabilities: S²Fs should be able to offload tasks to other S²F capable of handling the corresponding packets;
2. Support for intra- and inter-domain collaboration: collaboration among S²F should be allowed not only inside a same administrative domain, but also span multiple domains if desired;
3. Support for best effort collaboration: S²Fs should be able to engage in "best-effort" collaborations, treating only a fraction of the traffic to which some offloaded task applies;
4. Support for temporary collaboration: S²F should be able to establish temporary collaborations, which are associated to an agreed-upon validity period;
5. On demand security: the framework should allow S²Fs to be instantiated on demand, whenever necessary.

Complementary to such functional requirements, the following non-functional requirements are associated with the solution:

1. Quality of service for legitimate traffic;

2. Optimization of network resources and reduced packet latency.

As it should be clear from the requirements hereby presented, in this work we focus on collaboration aspects rather than addressing DDoS attack detection or mitigation mechanisms themselves. The motivation for this focus is that, as highlighted in recent surveys on this subject (*e.g.*, (SHAMELI-SENDI et al., 2015; MENG et al., 2015b)), many detection and/or mitigation solutions exist; however, an open challenge is exactly to enable collaboration among them, thus increasing detection accuracy and mitigation efficiency while optimizing resource consumption.

4.3 General Description

This work proposes a novel security architecture that is able to detect and mitigate DDoS attacks cooperatively through one or more network domains assuming an orchestrated architecture. Such orchestration is performed by a specialized central component responsible for establishing the negotiation between different S²Fs, allowing the definition of the amount of resources that must be outsourced. The effective attack mitigation is then performed considering different strategies that enable the best use of the available S²F resources through a security tasks offload. Those strategies may involve a best effort resource offloading, the application of a resource optimization algorithm that enables the optimal placement of security tasks or the combination of both.

During the collaboration process of the proposed architecture, although DDoS mitigation is performed cooperatively, the attack detection is performed individually by each S²F considering relevant variables being monitored, such as high packet dropping rate, packet latency or CPU usage. The main goal of the monitoring capability enabled in each node is to detect high usage of computational resources. Then, once the attack is

confirmed, the collaboration is launched through a negotiation between the centralized controller and each S²F that will help in the collaboration. In addition, when the available overall processing capacity is not enough to handle the security tasks offload even after the best effort and optimization strategies are launched, a new security node may be provisioned in order to handle the Initiator overflow.

4.4 Collaboration among Security Service Functions

In this section, we discuss the elements involved in the proposed collaboration framework, as well as their organization based on the SFC architecture. The SFC architecture is employed due to its flexibility in handling network packet flows through S²F, creating a chain of security services responsible for performing attack mitigation and/or detection. Such flexibility makes it possible to redirect part of the traffic, so it can be handled by nodes capable of treating suspicious traffic accordingly. The reason for using SFC is that, although alternatives exist (*e.g.*, a purely SDN-based approach as done in (SAHAY et al., 2015a)), SFC already provides many mechanisms that facilitate the construction and management of paths with different security services inside a domain.

4.4.1 Classes of Collaboration Elements

S²Fs and Controllers are the main pieces that compose the proposed framework. They are distributed among one or different network nodes composing a chain of services where each service represents a Firewall, a IDS or a IPS.

Taking into account the assumptions made in the previous paragraph, each element hereby considered can be grouped in the following types:

- **Initiator S²F:** Services that face congestion, resulting in bottlenecks that leads to the decrease in the QoS for a given targeted Service. Such bottlenecks

are identified through high packet dropping rates and computational resources insufficiency.

- **Provider S²F:** Services responsible for helping in the collaboration process. These services may be located at various places on the path and the distribution of security tasks among these S²Fs is performed by the centralized controller.

- **Controller:** The module located in each domain involved in the collaboration process that is responsible for handling the proposed collaboration strategies during a DDoS attack and for collecting relevant monitoring information of each enabled node located in its domain. Such monitoring information is particularly important to correctly fulfill the variables of the optimization algorithm.

4.5 Best-effort Filtering

The collaboration among S²Fs can be seen as an agreement on dedicated resources, allowing a cooperation to be established among those nodes in what we call a "best-effort" mode. The strategy provided by a best-effort mode allows that, whenever the amount of resources required to perform the outsourced task reaches the agreed threshold, the remaining traffic is not treated by the S²F that engaged in the collaboration. Instead, all untreated packets are marked so they can be treated later by other S²Fs that are able to handle more traffic than the amount previously negotiated with the controller. Such behavior is expected to enhance the collaboration process performance, as it adds flexibility to the system and prevents undesirable situations such as resource exhaustion or resource hijacking, which may occur in systems that only allow the complete outsourcing of security functions. Also, during a best-effort interaction, all participating S²Fs should be able to engage in temporary collaborations, treating a fraction of the traffic for an agreed period of time and relieving the packet treatment after the attack is gone and congestion is not detected anymore.

Such best-effort strategy is possible due to the flexibility of the SFC architecture when providing different paths to specific network flows. Specifically, it allows the definition of a separate datapath (*e.g.*, with VLAN tags) for a suspicious flow, which may include in all S²Fs required for its analysis (SAHAY et al., 2015a), for example. This work also aims at taking into account the load on the S²Fs in a path, provisioning new ones if the best-effort strategy is not enough to provide treatment to the whole malicious traffic.

In addition, the collaboration among elements in most existing solutions is usually of the type "all-or-nothing", as all traffic matching a given security policy must be fully handled by the node that accepts to perform a given task without considering resource usage issues. Instead, additional nodes allocation or load balancing are a more widely adopted strategy to deal with distributed attacks that cause congestion in the service chain. Such behavior is not necessarily optimal in terms of resource usage when compared to the approach hereby proposed, as further discussed in Chapter 5, leading to unnecessary overheads caused by additional resources allocation after a volumetric attack is detected.

4.6 Optimal Allocation of Security Tasks

Depending on how the security tasks are allocated along the network, a variety of optimization opportunities become available. Hence, it makes sense to search for optimal strategies, or at least avoid bottlenecks when placing Provider S²Fs for treating some packet flow.

When trying to cope with the inherent complexity of such tasks allocation optimization, existing collaborative solutions usually limit their scope to a subset of computing or network resources (e.g, CPU/RAM usage). Similarly, the proposed framework focus mainly on CPU usage, since this is the main parameter responsible for the queues'

growth on each S²F. Consequently, this metric is strongly correlated with degradation of service quality in terms of latency and packet dropping rates (BULLOT; COTTRELL; HUGHES-JONES, 2003). Nevertheless, the proposed optimization approach, based on Linear Integer Programming, should be easily adaptable for other metrics if desired.

Another important aspect when optimizing the distribution of security tasks is that the resource allocation needs to obey some security constraints. For example, as discussed in (SENDI et al., 2016), in some scenarios the most efficient allocation strategy might violate security best-practices; therefore, no strategy should be adopted blindly. Also, existing strategies that deal with resource optimization, such as (QAZI et al., 2013), are usually focused on finding the optimal location for the instantiation of *additional* specialized nodes in the service chain when dealing with a congestion, thus demanding even more resources. Conversely, the strategy hereby described focus on consuming resources already available, *i.e.*, it comprises an optimization step that comes before the allocation of new nodes. It is still more oriented toward prioritizing performance over security-related best practices, though, since its main goal is to improve resource usage on S²F chains (assumed to already deal with the underlying security concerns and eventual conflicts). Hence, our proposal is more related to approaches like the one described in (SAHHAF et al., 2015), which decomposes security policies and distributes them among specialized nodes in a service chain.

4.6.1 Integer Linear Programming Formulation

One common characteristic observed in solutions that optimize the resource usage, and also adopted in our proposal, relies on Integer Linear Programming (ILP) to model the optimization problem. Such optimization algorithm allows a variety of constraints to be combined, so the lowest or highest associated cost can be identified considering those constraints. Indeed, an ILP formulation is quite useful to address problems such as resource allocation, in particular due to its flexibility to deal with different variables

and constraints. In the particular case of our proposal, the ILP formulation aims at minimizing the CPU cost while allocating security rules in different positions in the S²F chain. At the same time, it defines suitable constraints for prioritizing the overall network performance.

4.6.1.1 Objective Function

The objective function formulated for the ILP problem considers the following variables:

- Rule weight (W): Each rule in this formulation has an associated weight, corresponding to the amount of computational load a given S²F is capable of handling. More precisely, the sum of all rules instantiated in a node cannot be larger than its computational capacity, and all rules aims at filtering only the traffic identified as suspicious.

- Computational load (L): Each S²F enabled in the protected domains is capable of handling the incoming traffic until the exhaustion of some of its computational resources. Soon after a given S²F becomes overloaded, part of the incoming packets passing through it would be dropped, negatively affecting the service that consumes these packets. Thus, each S²F is monitored trying to prevent the maximum allowed traffic amount from being reached. This is done even if the ILP formulation is not sufficient to redistribute the traffic appropriately among available nodes, in which case the provision of a new S²F in the chain becomes necessary. The computational load is, thus, the main variable in the proposed ILP formulation, since the minimum sum of loads of all enabled S²Fs in a given domain is the target of the proposed Objective Function.

- Node ordering variable (i): Each S²F in the formulation has an associated position in the security services chain. Such position is important to define the distribution

of tasks among the nodes, since the computational load associated to a rule is expected to be more significant the closer to the beginning of the chain that rule is applied. After all, filtering rules cause suspicious packets to be elliminated earlier in the service chain, so they do not reach the S²Fs positioned closer to the end of the chain.

- Rules ordering variable (j): Allows rules positioning in each node composing the S²F chain.

In an ILP formulation, the definition of adequate constraints is important to ensure that the optimal result considers all system variables. This affects the efficiency of the optimization. Considering the objective function and assumptions made in this sub-section, the following constraints are considered in this work:

- 0 < W < 100: The rule weight, which is limited by the computational load in a given S²F;
- 0 < L < 100: The traffic load is limited by the maximum amount of computational resources available at a given S²F;
- 0 < j < n: The number of rules enabled in a given S²F is limited by the amount of rules in the datapath;
- 0 < i < n: The number of positions that may be used to implement a given rule is limited by the number of S²Fs in the datapath;
- $0 < \sum_{j=1}^{n}(Wj) <$ L: The sum of rules located in a given S²F, also considering their respective weights, is limited by the S²F's available traffic load.

Considering the mentioned variables and assumptions, the Objective Function formulated for the ILP problem is written as:

$$\min \sum_{i=1}^{n} [Li - \sum_{j=1}^{n} (Wi, j)] * i$$

$$\sum_{j=1}^{j=n} (Wj) < L$$

$L, W \leq 100$

$L, W, R, i, j \in \mathbb{N}$

The proposed ILP formulation is employed in the proposed Collaborative Architecture as a module in the ODL controller through the use of the PuLP ILP solver for the Python programming language (MITCHELL et al., 2009). It must also be highlighted that, in order to enhance the resource optimization during any kind of volumetric DDoS attack, the ILP formulation may be combined with the best-effort capability described in Section 4.5.

5 EXPERIMENTAL ANALYSIS

Aiming at evaluating the feasibility of the presented work, an experimental setup that emulates the insertion of S²F instances in a chain was built, allowing the measurement of the benefits of the S²Fs' collaboration. For this, the SFC OpenDaylight (ODL) project in its Beryllium release (OpenDaylight, 2016) was used to provide the needed infrastructure for the ODL controller (Linux Foundation, 2015) to provision and control service chains. Then, a general testbed was built to be used as a basis for the analysis hereby described, even though in only a part of its elements and/or functionalities would be necessary (as detailed in each of the following subsections). Basically, it provides emulated forwarders (*i.e.*, SFF), in the form of Python applications that can communicate with the ODL controller and with any instantiated S²F defined as part of a chain. Using this infrastructure, the experimental testbed was then set up in a virtualized environment created with Vagrant (PALAT, 2012), VirtualBox (ORACLE, 2016), Open vSwitch (PFAFF et al., 2009).

The resulting testbed, illustrated in Figure 10, emulates the behavior of a service network composed of two network domains, representing a cloudlet and a cloud. Such domains are implemented in two physical servers, both having the following specification: Intel Xeon E5-2430 processor (15M Cache, 2.20 GHz), 48GB (6x8GB) of DDR3 RAM memory, and Intel Gigabit ET2 Quad Port Server, running GNU/Linux Ubuntu 14.04.4 LTS (GNU/Linux 3.13.0-24-generic x86_64). Each server runs VirtualBox (ORACLE, 2016) with a similar set of VMs: an SFC controller (`odl`) that acts as an orchestrator for the SFC infrastructure; two S²F, both able to emulate Deep Packet Ins-

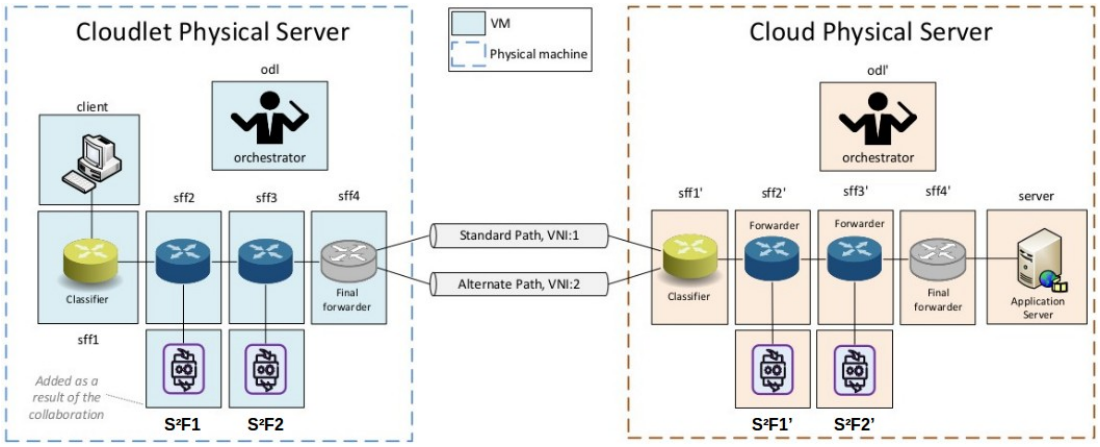

Figure 10: Overall SFC testbed employed in our experiments: communication between a client (JMeter) and a web server in two different domains.

pection (DPI) services by performing some CPU-intensive task, besides handling the packets' NSH so they can be correctly treated by the underlying forwarders; and four forwarders (numbered `sff1` to `sff4`), the first acting as a classifier at the beginning of the chain, two forwarding the traffic to the S²F in the middle of the chain, and the last one handling the Standard and Alternate paths at the chain's end, using different virtual network interfaces (VNI) for connecting to the cloud's Classifier. This scenario corresponds to a scenario post-negotiation between cloud and cloudlet, which may result, for example, in the instantiation of a new S²F on the latter's domain for running some task (in Figure 10, this is the case for `S²F1` on the cloudlet's side).

The traffic generation for the experiments was performed using a JMeter client in the cloudlet to send increasing amounts of HTTP requests toward a web server in the cloud. In all tests involving best-effort collaboration, we assume the branching approach depicted in Figure 11. Hence, the standard and alternate paths are configured as different SFC chains, and the collaboration's Provider reclassifies non-treated traffic so it is directed to the path including the Initiator.

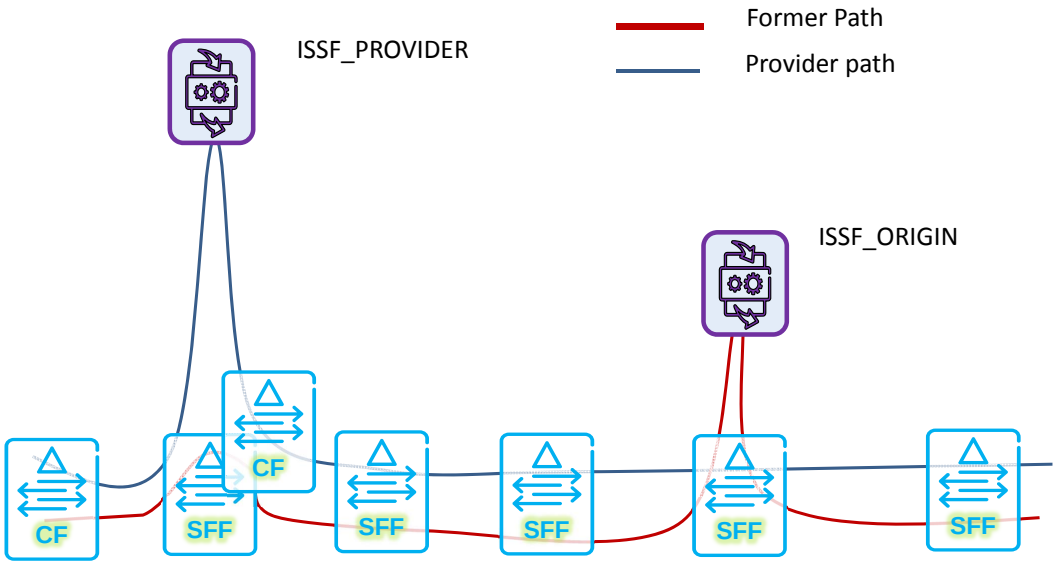

Figure 11: Latency incurred by SFC architecture, with one forwarder per S²F.

5.1 Adding a New S²F to the Service Chain

To assess the overhead introduced by the path modification mechanisms when using the branching approach, we measured the time necessary for adding `S²F1` into the Cloudlet's chain, which gives a general base level of the overhead incurred by any S²F insertion into a previously configured chain. This task was separated in two steps: (1) the time taken for instantiating a VM image; and (2) the time taken for creating a new SFC chain to which the already instantiated S²F is added. The result is that the first and second steps take, respectively, 64.9 ± 0.3 and 69.5 ± 0.1 seconds.

In practice, however, the first step would be executed only if the VM was instantiated on demand after the negotiation between orchestrators is finished; hence, such overhead could be neglected with the preemptive instantiation of a suitable VM as part of the collaboration protocol. Alternatively, this overhead could be reduced to less than 10 seconds if the VMs are resumed on-demand rather than instantiated from scratch (KNAUTH; FETZER, 2013), and would possibly be even lower in systems adopting a

more lightweight virtualization approach (*e.g.*, container-based (XAVIER et al., 2013)).

In addition, the second step's overhead can also be reduced through the preventive creation of Alternate paths inside cloudlet and cloud domains, based on regular DDoS occurrences or previous collaboration records with other domains. Such preventive paths and corresponding VMs may remain idle until a security incident is detected, when it is then activated for operating as an Alternate path. In this case, the overhead of the second step would be reduced to the time necessary to modify the classification rules inside the participant domains, which should take the order of 1 ms from the data plan's perspective (ROTSOS et al., 2012).

All in all, and even considering the worst case scenario where VMs and paths are instantiated on demand, the solution's total overhead remains around two minutes, which is reasonable when compared to the hours-long lifespan of typical DDoS attacks, as indicated in security reports provided by Akamai and others (AKAMAI, 2018; MATTHEWS, 2014; PESCATORE, 2014).

5.2 Measuring SFC's Latency Overhead

In a second experiment, the average overhead introduced by the adoption of SFC itself as basis for an orchestrated implementation of the proposed framework was measured. More precisely, it was measured how much latency is added by the presence of forwarders along the chain in comparison with having a direct connection between the S²F. For this task, it was employed the same S²F provided by default in the OpenDaylight SFC demo (OpenDaylight, 2016), which do not perform any actual service (*i.e.*, does not run a DPI-like task), but simply forwards all traffic received after evaluating and updating the packets' headers. The network interfaces of those S²Fs were then connected (1) via forwarders, one per S²F, analogously to the scenario shown in Figure 10, or (2) directly, so no overhead would result from the SFC architecture itself. Figure 12

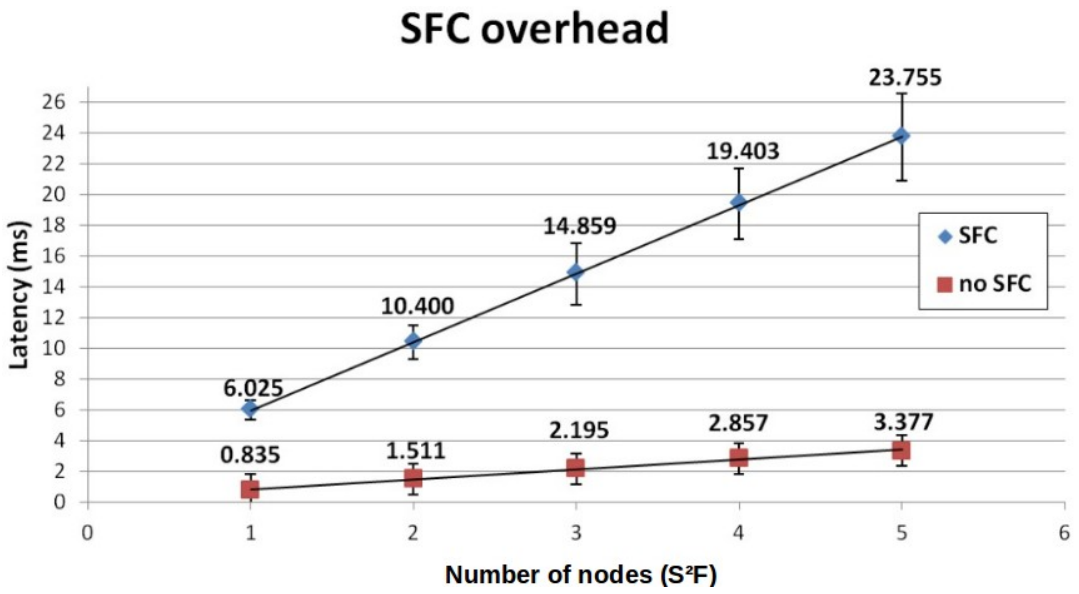

Figure 12: Latency incurred by SFC architecture, with one forwarder per S²F.

depicts the results obtained considering a total number of S²F ranging from 1 to 5, showing a linear growth with every additional S²F; the numbers obtained correspond to 1000 repetitions of the tests. The analysis of this figure reveals that each S²F and corresponding forwarder contribute to an average latency of approximately 4.5 ms, which means a total latency overhead of approximately 3.8 ms per S²F compared to the scenario where the S²F are connected directly.

5.3 Evaluating the Benefits of Best-effort Collaboration

In this section, the best-effort collaboration approach is analyzed aiming to identify how it affects the communication's QoS in terms of packet drop rate, latency and jitter. To avoid perturbations from other S²Fs, which could add noise to the measurements, `S²F1` and `S²F2` were removed from the testbed's chain and only `S²F1'` and `S²F2'` were left as the collaborations' Provider and Initiator, respectively. Both were then pre-configured (1) to perform a CPU-intensive task on the traffic coming from the cloudlet, thus emulating a DPI service for each packet it decides to treat, and (2) with different collaboration percentages, to represent distinct mitigation scenarios. For example, for

a 30% collaboration percentage, the Provider treats 30% of the incoming packets and sends them through the Standard path (which does not include the Initiator), whereas the remaining 70% of packets are simply reclassified and forwarded to the Alternate path to be treated by the Initiator; this applies only to the inbound traffic, since the cloud's web server's response packets are simply forwarded back to the client without any additional treatment. JMeter was then used to send an increasing amount of traffic to the server, following a linear growth rate, aiming to generate enough packets to overload the intermediary S²Fs and, hence, induce them to drop part of the traffic. As a result, Figure 13 depicts the percentage of packets that are dropped in this experiment for similar traffic loads, both in terms of total number of packets and of growth rate, as informed by JMeter.

It is possible to note that collaboration provides some interesting optimization opportunities: if the Provider is configured to treat packets until it starts being unable to do so without dropping packets, the network can considerably reduce packet drop statistics. Specifically in the case of this experiment, a direct result of collaboration is that the drop rate goes from 31% in the original scenario, in which the Initiator was handling all traffic, to 2.7% when 40% of the traffic is treated by the Provider. Thus, it is possible to affirm that the employment of a load balancer, which would split half of the traffic to be handled by the Provider, is less effective than a best-effort approach due to the flexibility in the amount of offloaded tasks according to available resources in each node. It is also important to note that, when the Provider handles more than 40% of traffic, it becomes itself overloaded, and starts dropping packets. In the worst case, when the Provider handles 100% of the traffic on behalf of the Initiator, the former ends up dropping approximately as many packets as before the collaboration was set (namely, 27%), showing that fully offloading the Initiator's tasks is far from being an optimal solution. Obviously, different settings may lead to different optimal points; for example, if the amount of idle resources at the collaborating VMs are

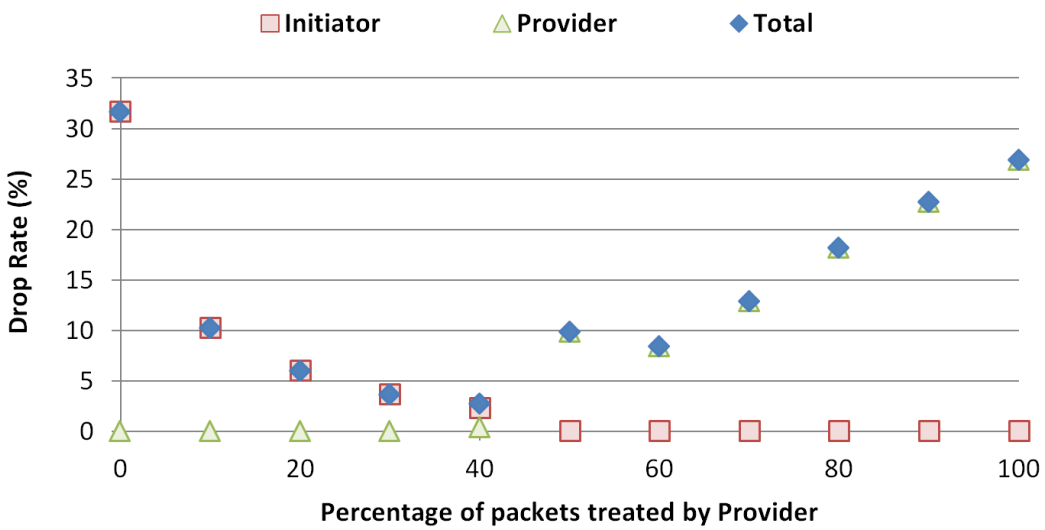

Figure 13: Packet drop rate for different collaboration percentages.

different, then the ideal collaboration percentage is likely to have a higher percentage at the less loaded VM. Nevertheless, we can conclude that the proposed framework successfully extends the overall security resources to the sum of available resources in the S²Fs participating in the collaboration. Then, this pool of resources can be taken into account by algorithms for optimizing the distribution of tasks, which (1) potentially avoids the need of instantiating copies of overloaded S²F for the purpose of load balancing, and (2) allows the different positions of S²F along the chain to be taken into account (as further evaluated in Section 5.4).

In the same setting, the behavior of the packet dropping ratio rate over time is also assessed as traffic grows. The results are shown in Figure 14. For better visibility, this figure only includes some selected collaboration percentages (namely, 0% to 100% in steps of 20%, besides the 50% inflection point), and starts at 120 seconds of simulation time (since, before that, no packet is dropped for any collaboration percentage). The resulting graphs once again indicate that, with a suitable configuration for the best-effort collaboration (optimally, at 40% in our setup), packet dropping take more time to occur when the network is submitted to similar traffic loads (or, equivalently, to equally sized volumetric DDoS attacks). This also means that the traffic losses happen for much

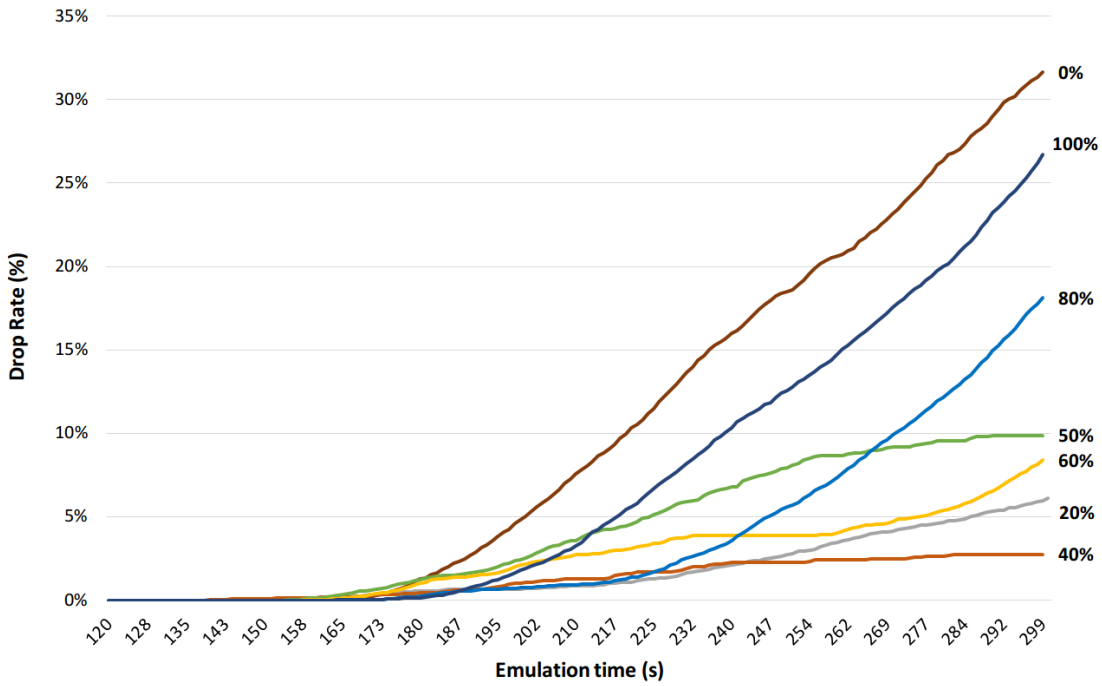

Figure 14: Packet drop rates during execution time, as traffic increases, for different percentages of traffic being handled by the Provider.

higher traffic, *i.e.*, that the framework can significantly improve the system's resilience against packets dropping during traffic bursts.

It is also possible to note that, even if Provider and Initiator are identical in terms of resource availability like in our testbed, collaboration approaches placing a higher load on the former tend to be less effective than those concentrating the burden on the latter. One probable reason is that even a small amount of packet treatment by the Provider serves to alleviate the burden on the Initiator, as the latter does not receive some packets. In comparison, the Provider always receives the full load of packets, independently of the subsequent treatment by the Initiator, so the overall result of overloading it is more dropped packets and less efficiency.

Using JMeter's measurements, it is possible to evaluate the benefits of best-effort collaboration in terms of latency and jitter. The results are shown in Figure 15, which compares the average latency considering the whole experiment with the one obtained before any packet is dropped, for different collaboration settings. This Figure shows

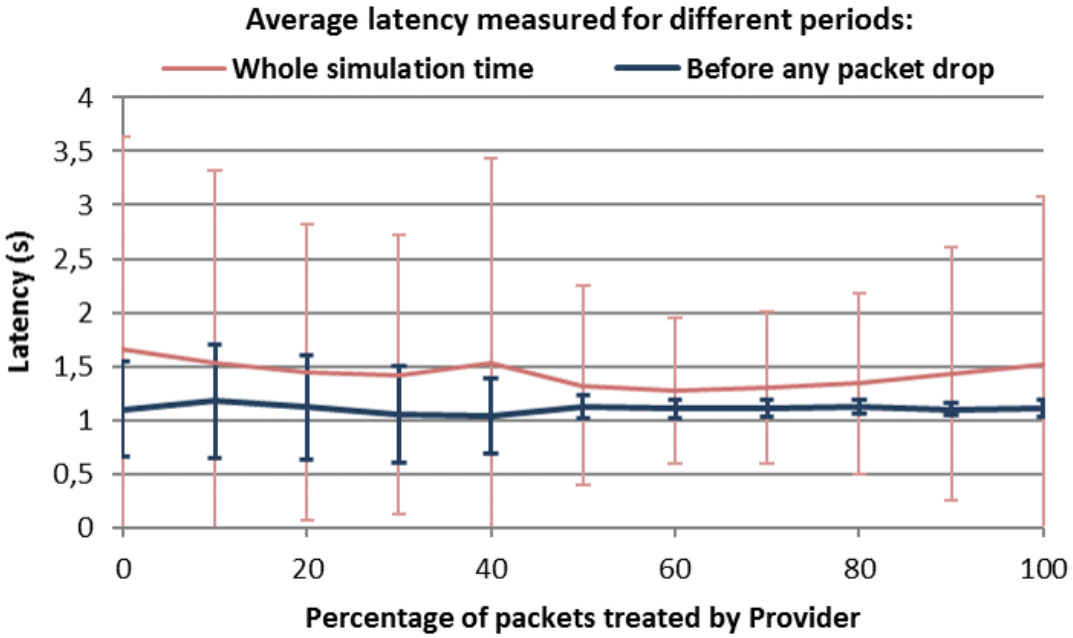

Figure 15: Latency and jitter for different collaboration percentages.

that the latency is on average 30% higher when we consider the period of time in which the network is overloaded and HTTP packets keep being dropped and retransmitted, while its standard deviation (and, thus, the network's jitter) is about 7x in this situation. Since, as previously discussed, the proposed framework delays such overloads as traffic grows, it also delays a rise in packet latency and jitter, improving the system's resilience to traffic bursts also in terms of these metrics. It is also interesting to notice that placing a higher burden on the Provider potentially leads to a lower jitter when the traffic is still under control (*i.e.*, while no packets are being lost). The reason is that, in this case, most traffic passes through only one S²F rather than two, so the packets are on average less prone to uncertainties caused by queuing on those nodes.

5.4 Assessing Packet Filtering in Different Points of the Chain

In the previous experiments, the S²Fs were configured to emulate a packet-inspection process, but would not filter any traffic, *i.e.*, all packets were considered

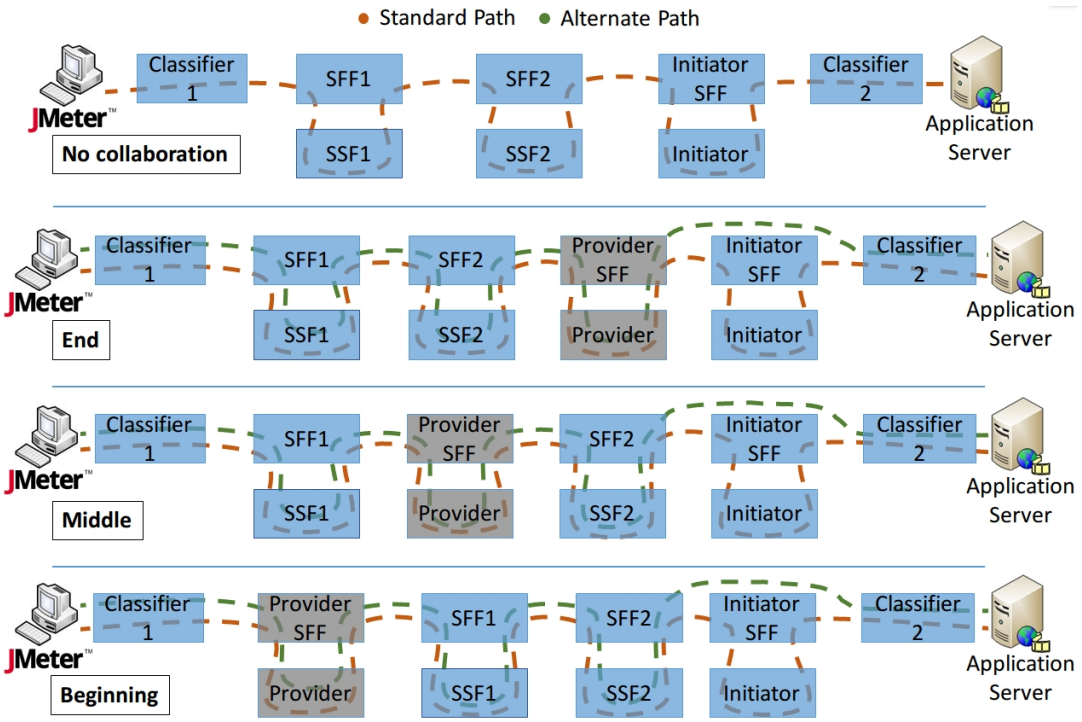

Figure 16: Collaboration with a provider placed at different points for filtering traffic.

legitimate and forwarded to the web server. In this subsection, we complement those experiments considering a scenario were 50% of all traffic arriving at the chain is malicious and, thus, is filtered by one of the S²F after the inspection process, either by itself or in collaboration with a Provider placed at different points of the chain. Specifically, we consider the four scenarios shown in Figure 16: a chain originally composed of three S²F, the last of which is the only one capable of filtering the malicious traffic; since it can become overloaded, a fourth S²F is introduced in different positions of the chain, collaborating with the Initiator by filtering 40% of the malicious traffic. This testbed is equivalent to the basic one shown in Figure 10, although the tests are performed in a single domain to avoid noise that might originate from the additional classifiers and tunnels connecting two domains.

The experiments performed aim to show the advantages of adopting a best-effort collaboration approach rather than regular load balancing mechanisms for lessening the burden at the Initiator, since the former allows attacks to be mitigated closer to their sources. In other words, by having the Provider filter packets earlier in the datapath

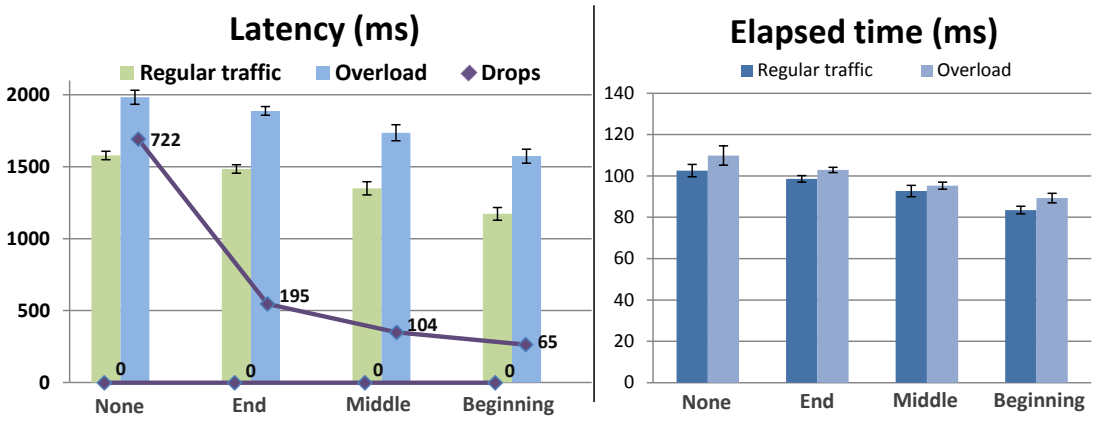

Figure 17: Latency, elapsed time and number of packets dropped when a Provider is inserted into different points of a chain originally composed by three S²F.

rather than at the same place as the Initiator, packets that would be dropped anyway due to network congestion are also removed earlier, enhancing the network's QoS. To test this, different loads of HTTP requests were sent via JMeter to a web server to allow the evaluation on how the Provider's logical position affects (1) the average latency for each request, (2) the total elapsed time to serve them all, and (3) the number of packets dropped during this time. Analogous to the experiments in Section 5.3, two traffic patterns are analyzed: regular traffic, sent in a rate low enough to prevent any packet from being dropped by the S²F along the chain; and overload, in which packets are expected to be dropped due to congestion at the S²F in response to a traffic burst. The method for obtaining those traffic patterns consists in configuring JMeter for running different numbers of simultaneous threads, but maintaining approximately unchanged the total number of requests made to allow a comparison between regular and burst traffic. For example, if 1000 HTTP requests are made, JMeter could be configured to run $n = 10$ simultaneous threads in a loop of size $\ell = 100$ for the regular traffic scenario, whereas the overload scenario could take $n = 20$ threads run $\ell = 50$ times. In order to avoid an exponential growth in the packet drop rate, however, the configurations actually adopted in our experiments were such that n and $n + 1$ were exactly around the point in which packets started being dropped in our testbed.

The average numbers obtained for 10 executions of each test, which led to a

standard deviation below 4%, are shown in Figure 17. This Figure shows that, as the Provider is moved closer to the beginning of the chain, we can see an improvement on all three metrics evaluated. Namely, compared to the scenario with no collaboration, placing the Provider close to the Initiator reduces network latency in 5% when the network is overloaded, and in 6% for regular traffic; placing it at the beginning of the chain, however, leads to a latency reduction of 21% and 26% for high and regular traffic, respectively. The percentage in which the total time required for handling all requests decreases follow a similar behavior, going from 4%–6% to around 18% when the Provider is moved from the end of the chain to its beggining, whereas the packet drop rate in the latter position is only 33% of what is obtained in the former. These results imply that, when the network is under heavy traffic, even though the burden of the Initiator could be lessened via load-balancing (which is comparable to placing the Provider at the end of the chain), the flexibility provided by the proposed best-effort collaboration mechanism makes it a more advantageous approach. More generally, even before the network starts dropping packets, the same flexibility can also be leveraged in collaborations between different domains for reducing service latency.

5.5 Assessing multiple filtering tasks allocation along the chain

In complement to Section 5.4, we also evaluated some scenarios in which there is more than one malicious flow in the system. Specifically, we consider a setup with (see Figure 18): three S²F, each having an identical initial load of either 25%, 50% or 75%; four sources of traffic, among which only one is legitimate; and three filtering tasks, one for each malicious traffic, each of which requiring a total CPU load of 25% on the node that executes it. In each scenario, we repeat the experiments from Section 5.4, running JMeter and evaluating the latency of each packet, the total time required for treating all of them, and the network's packet dropping rate. The distribution of the three tasks

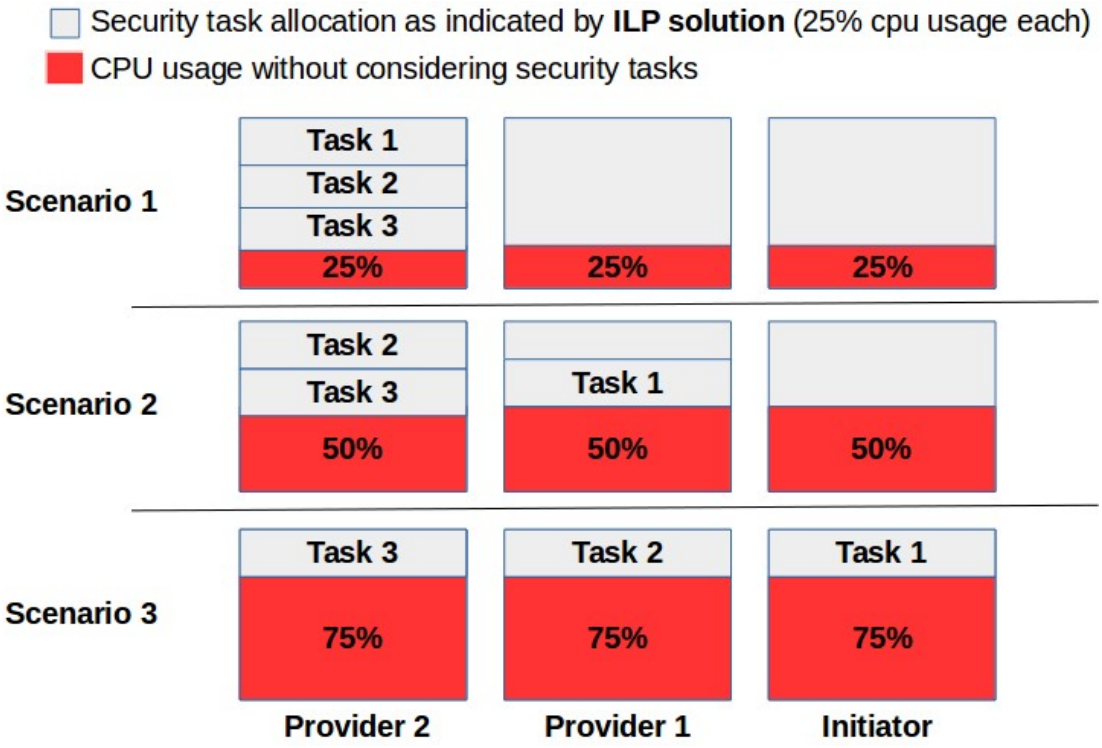

Figure 18: Packet filtering scenarios for multiple filtering tasks along the chain.

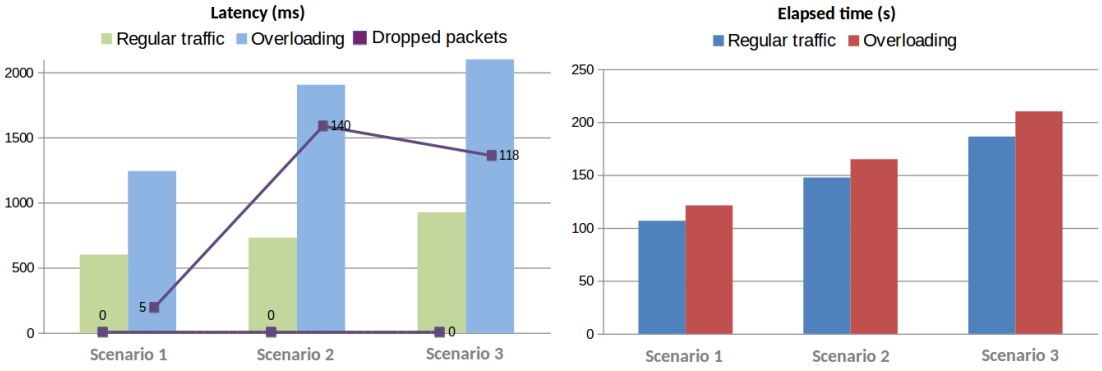

Figure 19: Packet filtering tasks allocation assessment results.

among the nodes follow the optimal distribution given by our ILP formulation (and confirmed in Section 5.4), meaning that tasks are concentrated earlier on the chain whenever possible.

The average numbers for 10 executions of each scenario are shown in Figure 19. Once again, we can observe that the allocation closer to the beginning of the chain leads to better results in terms of the evaluated metrics: when comparing Scenarios 1 and 3, it is possible to note a latency decrease of around 35% in the first case, where

all tasks are allocated at the beginning of the chain as defined by the ILP formulation under regular traffic. Considering the same comparison, the elapsed time to handle a thousand requests is decreased by 42.7%, a consequence of a reduced latency. In the cases where the chain is overloaded by malicious traffic, the difference is even bigger, as it is possible to note a latency decrease of around 49.3% in Scenario 1 when it is compared with Scenario 3, also allowing an elapsed time result 42.3 % smaller. Additionally to that, the total number of dropped packets is also considerably optimized in Scenario 1 compared to both Scenario 2 or 3. In the comparison between Scenario 1 and 2, for example, which presents the biggest difference in packet loss, the total dropped packets is 97% lower in the first case, evidencing the efficiency of the optimal allocation of security tasks in preserving legitimate traffic, as less legitimate packets are lost along the datapath.

6 CONCLUSION

In SECaaS (SECurity as a Service) systems, the effective collaboration among security service functions (S²F) is an important feature that allows a better scalability and resources usage. Indeed, existing works on collaborative security, most of which are dedicated to specific security elements (*e.g.*, firewalls and intrusion detection/prevention systems), have demonstrated that the collaboration among multiple security services enables more efficient detection and mitigation of attacks. With the network function virtualization trend, new opportunities appeared for managing fully virtualized appliances, as well as for enabling collaboration among them. This is the case of the recently proposed Service Function Chaining (SFC) architecture. Considering that, a flexible approach for enabling collaboration among multiple administrative domains that enables a best-effort and optimal task distribution is presented in this work.

The experiments based on the SFC architecture and OpenDaylight show that the proposed approach is feasible, incurs a low overhead, and enables an extension of the available resources for the security mechanisms to the sum of the resources available on the collaborating S²F instances. It also creates different optimization opportunities based on the placement of tasks through the S²F chain, providing a more flexible and efficient way of balancing the load among S²F in face of DDoS attacks while allowing the combination of more than one strategy. Such possibility allows the maximum possible usage of already available resources in the security services datapath, reducing latency and improving service quality for legitimate traffic.

6.1 Articles published and planned

The following publications, submissions and envisioned papers are a direct or indirect result of this research:

- **Performance Evaluation of IPsec Protocol in the Cloud using SR-IOV** (ALMEIDA et al., 2016). Conference paper presented at SBrT 2016, held in Santarém, Brazil;
- **Collaborative Security Architecture for Cloud Computing Systems** (ALMEIDA; SIMPLICIO, 2016). Conference poster presented at WPGEC 2016, held in São Paulo, Brazil;
- **A framework for enabling security services collaboration across multiple domains** (MIGAULT et al., 2017). Conference paper presented at ICDCS 2017, held in Atlanta, USA;
- **Demonstration of a framework for enabling security services collaboration across multiple domains** (ALMEIDA et al., 2018). Demonstration of the proposed framework presented at Netsoft 2018, held in Montreal, Canada;
- **A framework for enabling security services collaboration across multiple domains (Extended)** (MIGAULT et al., 2018). Journal paper published at Computers and Electrical Engineering journal.

6.2 Ideas for Future Work

The main future work envisioned for this research consists in the application of intelligent algorithms for predicting a Denial of Service behavior according to traffic monitoring data, enabling the collaboration among different network nodes before packets are dropped. Such capability would be useful for services that require high

resources availability, such as high quality video streaming or critical financial services running in cloud environments. Plugging such detection capability to the mitigation mechanism hereby described is expected to lead to a flexible and efficient security system for software-defined networking environments.

REFERENCES

AKAMAI. *State of the Internet / security – Q3 2017 executive review*. [S.l.], 2017. www.akamai.com/StateOfTheInternet.

____. *State of the Internet / security – Web attacks – Summer 2018*. [S.l.], 2018. www.akamai.com/StateOfTheInternet.

ALMEIDA, T.; SIMPLICIO, M. A. R. Collaborative security architecture for cloud computing systems. **V Workshop de Pós-Graduação - Engenharia de Computação - WPGEC 2016**. [S.l.], 2016. p. 13–16.

ALMEIDA, T. R.; ANDRADE, E. R.; BARROS, B. M.; JR, M. A. S. Avaliação de desempenho em nuvens computacionais utilizando ipsec em conjunto com sr-iov. 2016.

ALMEIDA, T. R.; BARROS, B. M.; ANDRADE, E. R.; SIMPLICIO, M. A.; MIGAULT, D.; POURZANDI, M. Demonstration of a framework for enabling security services collaboration across multiple domains. **2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)**. [S.l.], 2018. p. 341–343.

ANSTEE, D.; BOWEN, P.; CHUI, C.; SOCKRIDER, G. *Worldwide infrastructure security report*. [S.l.], 2016. XI. www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf.

ANTONAKAKIS, M.; APRIL, T.; BAILEY, M.; BERNHARD, M.; BURSZTEIN, E.; COCHRAN, J.; DURUMERIC, Z.; HALDERMAN, J. A.; INVERNIZZI, L.; KALLITSIS, M. et al. Understanding the mirai botnet. **USENIX Security Symposium**. [S.l.: s.n.], 2017. p. 1092–1110.

BJORKLUND, M. *Rfc 6020: Yang-a data modeling language for the network configuration protocol*. [S.l.]: October, 2010.

BRAUN, W.; MENTH, M. Software-defined networking using openflow: Protocols, applications and architectural design choices. *Future Internet*, Multidisciplinary Digital Publishing Institute, v. 6, n. 2, p. 302–336, 2014.

BULLOT, H.; COTTRELL, R. L.; HUGHES-JONES, R. Evaluation of advanced tcp stacks on fast long-distance production networks. *Journal of Grid Computing*, Springer, v. 1, n. 4, p. 345–359, 2003.

CHEN, X.; MU, B.; CHEN, Z. NetSecu: A collaborative network security platform for in-network security. **3rd Int. Conf. on Communications and Mobile Computing (CMC)**. [S.l.], 2011. p. 59–64.

CHESHIRE, S. *Hybrid Unicast/Multicast DNS-Based Service Discovery*. May 2017. https://tools.ietf.org/html/draft-ietf-dnssd-hybrid-05.

CHESHIRE, S.; KROCHMAL, M. *Multicast DNS (RFC 6762)*. feb 2013. https://tools.ietf.org/html/rfc6762.

____. *RFC 6763: DNS-based service discovery*. 2013. https://tools.ietf.org/html/rfc6763.

COMPAGNO, A.; CONTI, M.; GASTI, P.; TSUDIK, G. Poseidon: Mitigating interest flooding ddos attacks in named data networking. **Local Computer Networks (LCN), 2013 IEEE 38th Conference on**. [S.l.], 2013. p. 630–638.

CUPPENS, F.; CUPPENS-BOULAHIA, N.; GARCIA-ALFARO, J.; MOATAZ, T.; RIMASSON, X. Handling stateful firewall anomalies. In: *Information Security and Privacy Research*. [S.l.]: Springer, 2012. p. 174–186.

ENNS, R. *NETCONF configuration protocol*. [S.l.], 2006.

ENNS, R.; BJORKLUND, M.; SCHOENWAELDER, J.; BIERMAN, A. *RFC 6241 – Network Configuration Protocol (NETCONF)*. 2011. https://tools.ietf.org/html/rfc6241.

ERICKSON, D. The beacon openflow controller. **Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking**. [S.l.], 2013. p. 13–18.

FRANÇOIS, J.; AIB, I.; BOUTABA, R. Firecol: a collaborative protection network for the detection of flooding ddos attacks. *IEEE/ACM Transactions on Networking (TON)*, IEEE Press, v. 20, n. 6, p. 1828–1841, 2012.

GILLMAN, D.; LIN, Y.; MAGGS, B.; SITARAMAN, R. K. Protecting websites from attack with secure delivery networks. *Computer*, IEEE, v. 48, n. 4, p. 26–34, 2015.

GOUDA, M. G.; LIU, A. X. A model of stateful firewalls and its properties. **2005 International Conference on Dependable Systems and Networks (DSN'05)**. [S.l.], 2005. p. 128–137.

GREENE, K. TR10: Software-defined networking. *Technology Review (MIT)*, 2009.

GROUP, I. I. W. et al. *Interface to Network Security Functions (I2NSF)*. [S.l.]: accessed, 2017.

GUDE, N.; KOPONEN, T.; PETTIT, J.; PFAFF, B.; CASADO, M.; MCKEOWN, N.; SHENKER, S. Nox: towards an operating system for networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 3, p. 105–110, 2008.

HALPERN, J.; PIGNATARO, C. *RFC 7665 – Service function chaining (SFC) architecture*. 2015. https://tools.ietf.org/html/rfc7665.

HARRINGTON, D.; PRESUHN, R.; WIJNEN, B. *RFC 3411: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks*. 2002. Internet Engineering Task Force.

JINGLE, I. D. J.; RAJSINGH, E. B. ColShield: An effective and collaborative protection shield for the detection and prevention of collaborative flooding of DDoS attacks in wireless mesh networks. *Human-centric Computing and Information Sciences*, Springer, v. 4, n. 1, p. 1–19, 2014.

KIRKPATRICK, K. Software-defined networking. *Communications of the ACM*, ACM, v. 56, n. 9, p. 16–19, 2013.

KNAUTH, T.; FETZER, C. Fast virtual machine resume for agile cloud services. **2013 International Conference on Cloud and Green Computing**. [S.l.: s.n.], 2013. p. 127–134.

KREUTZ, D.; RAMOS, F. M. V.; VERÍSSIMO, P.; ROTHENBERG, C. E.; AZODOLMOLKY, S.; UHLIG, S. Software-defined networking: A comprehensive survey. *CoRR*, abs/1406.0440, 2014. Available at: <http://arxiv.org/abs/1406.0440>.

LEE, S. B.; KANG, M. S.; GLIGOR, V. D. CoDef: collaborative defense against large-scale link-flooding attacks. **Proc. of the 9th ACM conference on Emerging networking experiments and technologies**. [S.l.], 2013. p. 417–428.

LI, J.; BERG, S.; ZHANG, M.; REIHER, P.; WEI, T. Drawbridge: Software-defined DDoS-resistant traffic engineering. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 44, n. 4, p. 591–592, 2014. ISSN 0146-4833.

LIN, Y.; PITT, D.; HAUSHEER, D.; JOHNSON, E.; LIN, Y. Software-defined networking: Standardization for cloud computing's second wave. *Computer*, v. 47, n. 11, p. 19–21, 2014. ISSN 0018-9162.

Linux Foundation. *OpenDaylight, a Linux Foundation Collaborative Project*. 2015. http://www.opendaylight.org/. Accessed: 2016-05-01.

MAHAJAN, R.; BELLOVIN, S. M.; FLOYD, S.; IOANNIDIS, J.; PAXSON, V.; SHENKER, S. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM Computer Communication Review*, ACM, v. 32, n. 3, p. 62–73, 2002.

____. Controlling high bandwidth aggregates in the network. *ACM SIGCOMM Computer Communication Review*, ACM, v. 32, n. 3, p. 62–73, 2002.

MALIALIS, K.; KUDENKO, D. Large-scale ddos response using cooperative reinforcement learning. **11th European Workshop on Multi-Agent Systems (EUMAS)**. [S.l.: s.n.], 2013.

MALISHEVSKIY, A.; GURKAN, D.; DANE, L.; NARISETTY, R.; NARAYAN, S.; BAILEY, S. Openflow-based network management with visualization of managed elements. **Research and Educational Experiment Workshop (GREE), 2014 Third GENI**. [S.l.], 2014. p. 73–74.

MARQUES, P.; RASZUK, R.; MCPHERSON, D.; MAUCH, J.; GREENE, B.; SHETH, N. Dissemination of flow specification rules. *Arbor*, 2009.

MATTHEWS, T. *Incapsula Survey: What DDoS Attacks Really Cost Businesses*. [S.l.], 2014. http://lp.incapsula.com/ddos-impact-report.html.

MCKEOWN, N.; ANDERSON, T.; BALAKRISHNAN, H.; PARULKAR, G.; PETERSON, L.; REXFORD, J.; SHENKER, S.; TURNER, J. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008.

MEDVED, J.; VARGA, R.; TKACIK, A.; GRAY, K. OpenDaylight: Towards a Model-Driven SDN Controller architecture. **2014 IEEE 15th International Symposium on**. Sydney, Australia: IEEE, 2014. p. 1–6.

MENG, G.; LIU, Y.; ZHANG, J.; POKLUDA, A.; BOUTABA, R. Collaborative security: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, ACM, v. 48, n. 1, p. 1, 2015.

____. Collaborative security: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, ACM, v. 48, n. 1, p. 1, 2015.

MIGAULT, D.; SIMPLICIO, M. A.; BARROS, B. M.; POURZANDI, M.; ALMEIDA, T. R.; ANDRADE, E. R.; CARVALHO, T. C. A framework for enabling security services collaboration across multiple domains. **Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on**. [S.l.], 2017. p. 999–1010.

____. A framework for enabling security services collaboration across multiple domains. *Computers and Electrical Engineering*, Elsevier, p. 224–239, 2018.

MIJUMBI, R.; SERRAT, J.; GORRICHO, J.; BOUTEN, N.; TURCK, F.; BOUTABA, R. Network Function Virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys Tutorials*, v. 18, n. 1, p. 236–262, 2016. ISSN 1553-877X.

MIRKOVIĆ, J.; PRIER, G.; REIHER, P. Attacking ddos at the source. **Network Protocols, 2002. Proceedings. 10th IEEE International Conference on**. [S.l.], 2002. p. 312–321.

MIRKOVIC, J.; ROBINSON, M.; REIHER, P. Alliance formation for ddos defense. **Proceedings of the 2003 workshop on New security paradigms**. [S.l.], 2003. p. 11–18.

MITCHELL, S.; KEAN, A.; MASON, A.; O'SULLIVAN, M.; PHILLIPS, A. *Optimization with PuLP*. 2009. https://pythonhosted.org/PuLP/. Accessed: 2019-02-04.

MORROW, C.; DOBBINS, R. Ddos open threat signaling (dots) working group operational requirements. *IETF*, v. 93, 2015.

OIKONOMOU, G.; MIRKOVIC, J.; REIHER, P.; ROBINSON, M. A framework for a collaborative DDoS defense. **22nd Annual Computer Security Applications Conference (ACSAC'06)**. [S.l.], 2006. p. 33–42. ISSN 1063-9527.

OpenDaylight. *Service Function Chaining, The OpenDaylight Project*. 2016. https://wiki.opendaylight.org/view/Service_Function_Chaining:Main. Accessed: 2016-05-01.

OPENFLOW. *Specification, OpenFlow Switch – v1.3.0*. 2012.

ORACLE. *VirtualBox*. 2016. https://www.virtualbox.org/. Accessed: 2016-05-01.

PALAT, J. Introducing vagrant. *Linux Journal*, Belltown Media, v. 2012, n. 220, p. 2, 2012.

PAPADOPOULOS, C.; LINDELL, R.; MEHRINGER, J.; HUSSAIN, A.; GOVINDAN, R. Cossack: Coordinated suppression of simultaneous attacks. **null**. [S.l.], 2003. p. 2.

PESCATORE, J. *DDoS Attacks Advancing and Enduring: A SANS Survey*. [S.l.], 2014. https://www.sans.org/reading-room/whitepapers/analyst/ddos-attacks-advancing-enduring-survey-34700.

PFAFF, B.; DAVIE, B. The Open vSwitch Database Management Protocol. *IETF*, 2013. RFC.

PFAFF, B.; PETTIT, J.; AMIDON, K.; CASADO, M.; KOPONEN, T.; SHENKER, S. Extending networking into the virtualization layer. **Proc. of workshop on Hot Topics in Networks (Hotnets'09)**. [S.l.: s.n.], 2009.

QAZI, Z. A.; TU, C.-C.; CHIANG, L.; MIAO, R.; SEKAR, V.; YU, M. Simple-fying middlebox policy enforcement using sdn. *ACM SIGCOMM computer communication review*, ACM, v. 43, n. 4, p. 27–38, 2013.

REKHTER, Y.; LI, T.; HARES, S. *RFC 4271: A Border Gateway Protocol 4 (BGP-4)*. 2014. Internet Engineering Task Force.

ROTSOS, C.; SARRAR, N.; UHLIG, S.; SHERWOOD, R.; MOORE, A. W. Oflops: An open framework for openflow switch evaluation. In: ____. *Passive and Active Measurement: 13th International Conference, PAM 2012, Vienna, Austria, March 12-14th, 2012. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 85–95. ISBN 978-3-642-28537-0.

RYU, P. *Ryu sdn framework using openflow 1.3. Website*. 2014.

SAHAY, R.; BLANC, G.; ZHANG, Z.; DEBAR, H. Towards autonomic ddos mitigation using software defined networking. **NDSS Workshop on Security of Emerging Networking Technologies (SENT'2015)**. [S.l.], 2015.

____. Towards autonomic ddos mitigation using software defined networking. **SENT 2015: NDSS Workshop on Security of Emerging Networking Technologies**. [S.l.], 2015.

SAHHAF, S.; TAVERNIER, W.; ROST, M.; SCHMID, S.; COLLE, D.; PICKAVET, M.; DEMEESTER, P. Network service chaining with optimized network function embedding supporting service decompositions. *Computer Networks*, Elsevier, v. 93, p. 492–505, 2015.

SENDI, A. S.; JARRAYA, Y.; POURZANDI, M.; CHERIET, M. Efficient provisioning of security service function chaining using network security defense patterns. *IEEE Transactions on Services Computing*, IEEE, 2016.

SHAMELI-SENDI, A.; POURZANDI, M.; FEKIH-AHMED, M.; CHERIET, M. Taxonomy of distributed denial of service mitigation approaches for cloud computing. *Journal of Network and Computer Applications*, Elsevier, v. 58, p. 165–179, 2015.

WANG, Y.; ZHANG, Y.; SINGH, V.; LUMEZANU, C.; JIANG, G. NetFuse: Short-circuiting traffic surges in the cloud. **Communications (ICC), 2013 IEEE International Conference on**. [S.l.], 2013. p. 3514–3518.

XAVIER, M.; NEVES, M.; ROSSI, F.; FERRETO, T.; LANGE, T.; ROSE, C. Performance evaluation of container-based virtualization for high performance computing environments. **21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing**. [S.l.: s.n.], 2013. p. 233–240. ISSN 1066-6192.

YOU, Y.; ZULKERNINE, M.; HAQUE, A. A distributed defense framework for flooding-based DDoS attacks. **3rd International Conference on Availability, Reliability and Security (ARES'08)**. [S.l.], 2008. p. 245–252.

ZARGAR, S. T.; JOSHI, J. DiCoDefense: distributed collaborative defense against ddos flooding attacks. **34th IEEE Symposium on Security and Privacy (S&P'13) (Poster)**. [S.l.: s.n.], 2013.

ZARGAR, S. T.; JOSHI, J. B. A collaborative approach to facilitate intrusion detection and response against DDoS attacks. **Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), 2010 6th International Conference on**. [S.l.], 2010. p. 1–8.

ZHANG, G.; PARASHAR, M. Cooperative defence against ddos attacks. *Journal of Research and Practice in Information Technology*, Sydney, Australia: Australian Computer Society, c2000-, v. 38, n. 1, p. 69–84, 2006.