Using network resources to mitigate volumetric DDoS

Pierre-Edouard Fabre

THÈSE DE DOCTORAT DE TELECOM SUDPARIS

Spécialité : Informatique et Réseaux

École doctorale : Informatique, Télécommunications et Électronique de Paris

Présentée par

Pierre-Edouard FABRE

Pour obtenir le grade de

Docteur de Telecom SudParis

Utiliser les ressources réseaux pour atténuer les attaques DDoS volumétriques

---

Directeur de thèse : **Hervé DEBAR**

co-Encadrant de thèse : **Jouni VIINIKKA**

co-Encadrant de thèse : **Gregory BLANC**

soutenue le 13 Décembre 2018

Composition du Jury

Rapporteurs :
- Isabelle Chrisment, Professeure, Télécom Nancy, Université de Lorraine, France
- Guillaume Urvoy-Keller, Professeur, Université Nice Sophia Antipolis, France

Examinateurs :
- Bruno Defude, Professeur, Télécom SudParis, France
- Guillaume Doyen, Maître de Conférences, Université de Technologie de Troyes, France
- Hervé Debar, Professeur, Télécom SudParis, France
- Jouni Viinikka, Docteur, 6cure SAS, France

Invité :
- Gregory Blanc, Maître de Conférences, Télécom SudParis, France

n° NNT: 2018TELE0020

PHD THESIS TELECOM SUDPARIS

Speciality: Informatics and Networks

Doctoral School: Informatique, Télécommunications et Électronique de Paris

Presented by

Pierre-Edouard FABRE

To obtain the degree of

Doctor of Telecom SudParis

Using network resources to mitigate volumetric DDoS

Thesis Director: **Hervé DEBAR**

Thesis co-Advisor: **Jouni VIINIKKA**

Thesis co-Advisor: **Gregory BLANC**

Presented on December 13^{th}, 2018

Members of Jury

Reporters:
- Isabelle Chrisment, Professor, Télécom Nancy, Université de Lorraine, France
- Guillaume Urvoy-Keller, Professor, Université Nice Sophia Antipolis, France

Examiners:
- Bruno Defude, Professor, Télécom SudParis, France
- Guillaume Doyen, Lecturer, Université de Technologie de Troyes, France
- Hervé Debar, Professor, Télécom SudParis, France
- Jouni Viinikka, Doctor, 6cure SAS, France

Guest:
- Gregory Blanc, Lecturer, Télécom SudParis, France

n° NNT: 2018TELE0020

Abstract

Massive Denial of Service attacks represent a genuine threat for Internet service, but also significantly impact network service providers and even threat the Internet stability. There is a pressing need to control damages caused by such attacks.

Numerous works have been carried out, but were unable to combine the need for mitigation, the obligation to provide continuity of service and network constraints. Proposed countermeasures focus on authenticating legitimate traffic, filtering malicious traffic, making better use of interconnection between network equipment or absorbing attack with the help of available resources.

In this thesis, we propose a damage control mechanism against volumetric Denial of Services. Based on a novel attack signature and with the help of Multiprotocol Label Switching (MPLS) network functions, we isolate malicious from legitimate traffic. We apply a constraint-based forwarding to malicious traffic. The goals is to discard enough attack traffic to sustain network stability while preserving legitimate traffic. It is not only aware of attack details but also network resource, especially available bandwidth.

Following that network operators do not have equal visibility on their network, we also study the impact of operational constraints on the efficiency of a commonly recommended countermeasure, namely blacklist filtering. The operational criteria are the level of information about the attack and about the traffic inside the network. We then formulate scenario which operators can identify with. We demonstrate that the blacklist generation algorithm should be carefully chosen to fit the operator context while maximizing the filtering efficiency.

Résumé

Les attaques massives par déni de service représentent une menace pour les services Internet. Ils impactent aussi les fournisseurs de service réseau et menace même la stabilité de l'Internet. Il y a donc un besoin pressant de contrôler les dommages causés par ces attaques.

De nombreuses recherches ont été menées, mais aucune n'a été capable de combiner le besoin d'atténuation de l'attaque, avec l'obligation de continuité de service et les contraintes réseau. Les contre mesures proposées portent sur l'authentification des clients légitimes, le filtrage du trafic malicieux, une utilisation efficace des interconnections entre les équipements réseaux, ou l'absorption de l'attaque par les ressources disponibles.

Dans cette thèse, nous proposons un mécanisme de contrôle de dommages. Basé sur une nouvelle signature d'attaque et les fonctions réseaux du standard Multiprotocol Label Switching (MPLS), nous isolons le trafic malicieux du trafic légitime et appliquons des contraintes sur la transmission du trafic malicieux. Le but est de rejeter suffisamment de trafic d'attaque pour maintenir la stabilité du réseau tout en préservant le trafic légitime. La solution prend en compte des informations sur l'attaque, mais aussi les ressources réseaux.

Considérant que les opérateurs réseaux n'ont pas une même visibilité sur leur réseau, nous étudions l'impact de contraintes opérationnelles sur l'efficacité d'une contre mesure régulièrement recommandée, le filtrage par liste noire. Les critères d'évaluation sont le niveau d'information sur l'attaque ainsi que sur le trafic réseau. Nous formulons des scénarios auxquels chaque opérateur peut s'identifier. Nous démontrons que la l'algorithme de génération des listes noires doit être choisi avec précaution afin de maximiser l'efficacité du filtrage.

Acknowledgements

First, I would like to thank my supervisor Prof. Hervé Debar for his support and guidance. I deeply appreciate the valuable time he always devoted to me, despite his extremely busy schedule. I would like to thank my advisor Gregory Blanc, for his extensive comments on my work. His rigor has been a great value in this work.

I express my gratitude to my manager and thesis instructor Jouni Viinikka for his time even personal, his insightful comments, encouragement and moral support throughout this work. I really appreciate the trust Jouni had in me by making me work on new topics. You will always be a model of intellectual integrity and perseverance to me.

I am also thankful to the whole 6cure team, starting with Fabrice Clerc for his trust since my last internship and for giving me the opportunity to grow within the company. Thanks to Vincent for his continuous help and understanding. You have the biggest heart. Thanks to the current colleagues: Justine, Emmanuel, Françoise, Antoine R. and Antoine P., Adrien, Baptiste, Ghislain, Frederic and to a former one : Quentin. It is a great privilege to work with each of you, you make working fun.

I owe my thanks to my parents for their endless support throughout my life and for their understanding of my decisions which sometimes induced sacrifices. I give a distinctive thank to my family: Papa D, Mama F, Etienne, Aurélie, Yaya, Alexandre, Fis, Grace, René, Israël and my nephew Noah for your patience, moral support and your kindness. I also think of my grandmother who would have been proud of this accomplishment.

Finally, my deepest gratitude goes to my lovely wife and best friend Bénédicte for your patience with my homework and continuous support even during difficult times. Finishing this work would not have been possible without you. Also, I give a very special loving thank to my sweet son who challenged me at the end of my Ph.D work.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Contents

1.1 Motivation and Challenges

Network services are essential in the Internet, as they are the basis of almost all interactions. Their availability is, however, threaten by Denial of Service (DoS) attacks. Since 1974, which has seen the first DoS attack, DoS attacks have gained in strength, sophistication, complexity and frequency. Mitigating such attacks and maintaining the availability of services is complex and sometimes requires actions upstream from the service, as the victim does not always have the resources to mitigate the attack. For example, massive DoS attacks, which involve huge amount of malicious traffic, can require large resources from upstream network providers to mitigate the attack.

Denial of Service attacks do not only affect the targeted service, but also impact other Internet stakeholders. That is especially true for volumetric Distributed Denial of Service attacks, which disrupt a service by causing congestion upstream from the service. The huge volume of traffic leveraged by attackers in recent attacks (up to Terabits per second in 2016 [1]) endanger the stability of networks and also threaten the availability of all services using these networks. As demonstrated in these attacks the Internet of Things (IoT) paradigm paves the ways to even more powerful attacks. Mitigation mechanisms have then to scale to the volume of such attacks, such that network stability and network traffic forwarding are not impacted.

As a consequence, from the point of view of service or network provider, mitigation should prevent the spread of the treat to other customers or services. In other words, mitigation should allow to control damages caused to 1) the network; 2) the other services built upon the impacted network, as it will also induce disruption for those services; 3) the legitimate customers of the targeted service which cannot access to the targeted service. The challenge is then summarized as follows: mitigation has to clean attack traffic in such a manner that network behavior is not altered, while it preserves legitimate traffic.

It is then crucial for service providers to benefit from relevant mitigation mechanisms integrated within their networks. It then has to interoperate with existing infrastructure, where interactions are twofold. At first, mitigation retrieves contextual information about the situation, *e.g.* the attack details, so that it adjusts to the resources available within the network. Second, mitigation may require modification of networking functions, such as routing.

Past experience has shown that contextual information is dynamic, as for example, attack sources may shift along the attack timeline. A mitigation that is unaware of those changes is likely to apply outdated actions or configurations to the traffic. Such obsolete reactions can worsen the situation

We can them depicts the challenges as following:

- mitigation should address the specific needs of volumetric DDoS attacks attacks, that is the processing of high volume of malicious traffic;
- mitigation should prevent damages caused to infrastructure and allows legitimate clients to access service;
- mitigation should integrate and inter-operate with an existing operational network infrastructure;
- mitigation should be kept up-to-date with the attack traffic.

1.2 Hypothesis and Objectives

In order to contribute to the volumetric DDoS attacks mitigation area, we made the following assumptions. Some contributions have their own hypotheses, which are described in each chapter.

- A security component is responsible for the detection of attack vectors. The identification of attack flows is made available to the mitigation mechanisms through alert messages.
- Alert messages contain details about detected attack flows, such as flow telemetries.

- Network monitoring mechanisms provides information about the overall traffic inside the network. This information encompasses the identification of traffic as well as telemetries.

These hypotheses may seem, at first sight, strong. We expect, at least, to reasonably meet the first hypothesis. The last two ones are optional. We discuss in detail those three hypotheses in the Chapter 4.

The following objectives have been proposed to achieve the challenges we described in the previous section.

Objective A We propose a mitigation technique that make use of the network equipment capabilities, such that we benefit from the existing infrastructure resources and we can interact with them.

Objective B We study the ability to control damages caused by the attack, both in terms of filtered attack traffic and preserved legitimate traffic, of mitigation with regards of the operational constraints.

1.3 Contributions

Our proposed technique relies on multipath routing. Both legitimate and attack traffic are forwarded through different logical of physical path throughout the network. Mitigation is achieved by applying a differentiated quality of service treatment to each type of traffic and by prioritizing the and prioritizing the forwarding of legitimate traffic. We proposed a load-balancing based malicious traffic isolation technique. Build on top of Multiprotocol Label Switching (MPLS) network standard, we benefit from his network management and quality of services extension, namely Traffic Engineering and Differentiated Services. We propose the Mitigation Label, a special purpose MPLS label, which carry a signature of a Distributed Denial of Service attack. Malicious traffic is isolated in function of the Mitigation Label. The resulting technique allows network operator to segregate legitimate from malicious traffic using network forwarding function, *i.e.* without the overhead of traffic filtering.

We also study the efficiency of blacklists, a commonly advertised technique against volumetric distributed denial of service. We formalize the operator's context, *i.e.* the visibility he has on its network. The visibility refers to the amount on details the operator is able to gather about the overall traffic inside his network and particularly the malicious traffic. We study how such parameters impact the blacklist ability to successfully filter malicious traffic and to preserve legitimate traffic.

The contributions of this dissertation are summarized thereafter.

- We provide a study of the ability of existing load-balancing techniques to segregate malicious traffic from legitimate one.

- We propose a new load-balancing technique build on top of MPLS which allow isolating malicious traffic.

- We formalize the visibility of an operator on his network and study the efficiency of blacklists according to this model.

1.4 Outline of the Dissertation

The remainder of this dissertation is organized as follows:

Chapter 2 - State of the Art This chapter introduces the problem of Denial of Service (DoS) attacks. We briefly draw a review of DoS attacks and their characteristics. We then introduce the volumetric DDoS attacks area with regard of these aspects, namely the victim, the exploit leveraged to disrupt the service and the sources of the attack. We explore the metrics proposed by the literature to assess the impact of these attacks. This chapter present a state of the art on the Denial of Service attacks mitigation. We classify mitigation techniques in three categories: 1) admission control techniques, which decide whether a part of network traffic should pass or not; 2) detour-based mechanisms which benefit from the routing techniques; 3) absorbing mechanisms which aim at handling attack using existing resources. Lastly, we introduce our works by comparing it to existing techniques.

Chapter 3 - Load-Balancing based Mitigation Technique This chapter introduce our mitigation approach. Based on load-balancing and multipath routing, we aim at isolating suspicious traffic and applying an unprioritized forwarding scheme to it. We study the ability of a Cisco router to achieve attack traffic isolation using load-balancing mechanisms. We also review the use of a literature advocated load-balancing algorithm, which is based on the CRC16 hash function, to perform the traffic segregation. Noting the inability of the existing load-balancing schemes to split both legitimate and malicious traffic, we propose a new mechanism. Our load-balancing technique consist of a signature of the attack which is carried in an MPLS label. We experiment to confirm the ability to segregate traffic according to legitimacy and compare results to the literature.

Chapter 4 - Enhancing Blacklist-based Mitigation This chapter deals with filtering techniques based on blacklists. We introduce the generation problem of such mitigation, *i.e.* maximizing filtered attack traffic while reducing collateral damages and all with careful regard to the available inputs. We then formalize scenario to evaluate the efficiency of blacklist

in terms of filtered attack and legitimate traffic, with regards of the available data inputs, namely the identification and details of attack traffic and information about the overall traffic inside the network. We design simple blacklist generation algorithm that takes as input the information available. The efficiency of these functions is tested in experiments.

Chapter 2

State of the Art

Contents

This chapter first describes the Denial of Service security problem that this dissertation addresses. Then it describes network and security concepts applied to this problem by providing a review of literature on the detection methods as well mitigation methods such as filtering or effect alleviating. We then present metrics proposed by the literature to assess the impact of DDoS attacks.

We then review in details mitigation techniques for Denial of Service attacks. Mitigation mechanisms are divided into four categories according to how it prevents damages caused by the attack, as shown in Figure 2.1. Admission control mechanisms depicted in Section 2.3 focus on determining which part of traffic (*e.g.*, at the packet level or at the stream flow level) is permitted to flow inside the network. The mechanism can require traffic modifications (*cf.*, active mitigation) or not, *i.e.* it only uses the current characteristics of the traffic (*cf.*, passive mitigation). Detour-based mechanisms (*cf.*, Section 2.4) are often used in conjunction with another type of mitigation. They aim at reducing the DDoS congestion effect by taking advantage of the network meshing. Absorbing (*cf.*, Section 2.5) maintains the availability of the network by dampening or dismissing congestion effects at all costs. While some detouring mechanisms may serve a similar purpose, absorbing-based mitigations differ in that they are not based on path reconfiguration.

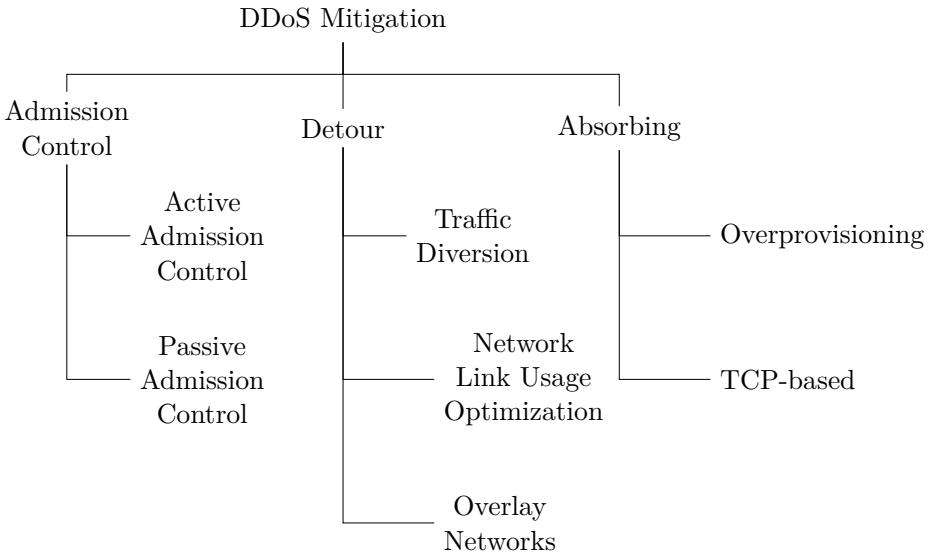


Figure 2.1 – DDoS mitigation categories

2.1 Denial of Service and the Internet

This section first introduces the Denial of Service (*DoS*) problem. It then reviews the different taxonomies of DoS attacks proposed by researchers to characterize attacks.

2.1.1 Introduction to Denial of Service attacks

As a part of information technology, the Internet is subject to the same fundamental security properties, namely *Confidentiality*, *Integrity* and *Availability* [2]. Each of the three terms defines a part of the attack surface. Confidentiality characterizes the fact that an information is not being viewed or accessed by unauthorized people. Integrity describes the capability to prevent an information from corruption or from being modified without authorization. Availability reflects the fact that authorized users can access information when they need it. In this respect, Denial of Service (*DoS*) attacks are specifically and solely targeting the Availability property by trying to disrupt a service provided in the Internet.

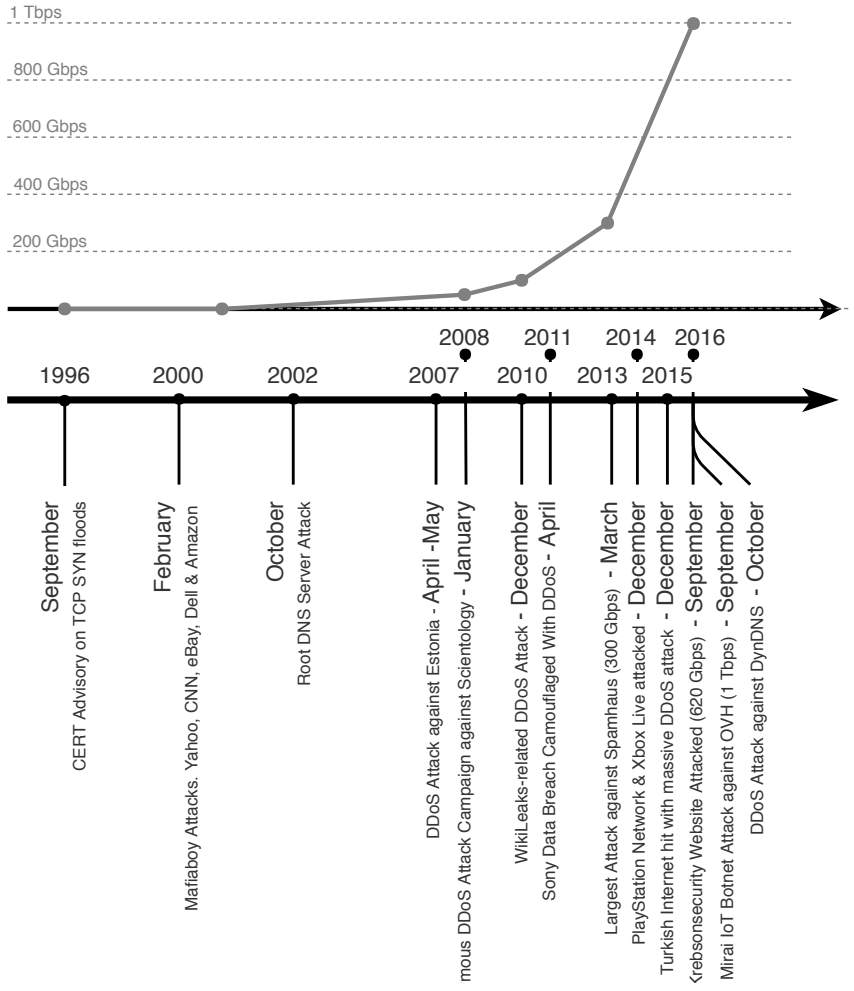


Figure 2.2 – Denial of Service Timeline

Concern about DoS attacks increased over the years due to the increase of number, frequency, scale, criticality and the decrease of required knowledge

and needed resources to trigger such an attack. It then became, in recent years one of the top threats of the Internet as shown in the DDoS timeline (Figure 2.2). Data required to draw this timeline is extracted from [3]. The increasing frequency goes hand in hand with the popularization of the concept of DoS as a Service (*DoSaaS*). It refers to websites that rent attack resources for a few minutes or hours. As a consequence, for a few dollars, people without network or security knowledge are able to launch such attacks. Figure 2.2 also demonstrates the growth of traffic volume (expressed in bytes/seconds) as new records are regularly established. For example, volume records reach 300 Gbps in 2013 and expand to successively 680 Gbps [4] and 1 Tbps [1] in 2016. A recent survey [5] stated that in 2015, half of data center respondents suffer from DoS attacks that completely saturated their Internet connectivity. In 2016, the emergence of Internet of Things (*IoT*) based botnet, *i.e.* network of devices known as bots and controlled by an attacker, opened the way for more massive attacks as 1) the number of potential bots remarkably grows due to the lack of security enforcement within this kind of devices, and 2) the required and available bandwidth to devices such as webcams is considerable. Beyond the service disruption caused to the targeted Internet service, it may also endanger the multiple Internet players such as Internet service providers, data centers and causes damage to other service. For example, in 2016, the attack against the Dyn DNS service provider affected the availability of some well-known services such as Netflix, Spotify, Twitter, Paypal, Amazon or Ebay. However, DoS attacks do not always induce a large volume of traffic. The Arbor Networks survey [5] shows that in 2015, 74% of attacks had a relatively small volume, *i.e.* 500Mbps, which is significantly less than the peak of 1 Tbps reached in 2016. Nonetheless, attacks with such smaller volume still threaten the availability of unprotected or inadequately protected targets.

It is worth noting that these high-volume peaks refer to the cumulated throughput of attack traffic at each target point of presence on the Internet. In fact, the target service may be hosted on several geographical nodes (*e.g.*, for enhancing its delivery), so that the attack can hit each node. The strike force would be more damaging if the attack concentrated on a single point.

2.1.2 DoS Attack Characteristics

Literature characterizes multiple aspects of Denial of Service attacks, such as attack sources, or means leveraged to achieve the service disruption. While some works provide a simple review of attack vectors such as protocol used and a list of common past and present attacks, some other works contribute to a better understanding of the problem by identifying and describing DoS attack characteristics.

Although we address a volumetric attack, *i.e.* a specific class of DoS attacks, a general review of the attack landscape allows us to enrich our

understanding of massive DoS attack distinctiveness. We start by reviewing exploits used by an attacker to achieve the service disruption. We then study attack sources and their capabilities, *i.e.* which exploit are available to them. Afterwards, we survey attack traffic details such as its rate dynamic or content validity. Finally, we provide a comprehensive characterization of existing attacks.

Victim

This attack does not only affect the final service, but multiple actors can also be hit and suffer from the effect of the attack.

Mirkovic and Reiher [6] differentiate between several victim types from most limited to important impact, namely application, host, resource, network and infrastructure. Application attacks only impact the victim's application. In that case, other applications hosted by the same host are not impacted. Conversely, attacks against the host completely disable the host by consuming its resources, leading to a disruption of all services located on the host machine. Resource-depleting attacks target software (*e.g.* DNS server) or hardware (*e.g.* router) critical resources in a network. Bandwidth-depletion attack names attack that consumes the bandwidth of network upstream from the targeted service. It mostly benefits from the funneling effect of convergence of traffic towards the final destination. Finally, infrastructure attacks refer to attacks that target some critical Internet services, such as DNS servers, routing protocols, *etc.* This last category reflects the attack synchronization of several attacks whose victim can be classified in another category, *e.g.* resource-depleting attacks.

Methods of Attack

Attack mechanism is closely related to the impact of attack as the different mechanisms are going to cause different effects.

Douligeris and Mitrokotsa [7] correlate the attack manifestation with the mechanism leveraged to disrupt the service. According to them the attack bandwidth or other resources depletion is achieved in different ways. On the one hand, the bandwidth is consumed by sending a massive amount of traffic either directly or indirectly to the target. On the other hand, other resources are depleted either by exploiting network protocol design weaknesses or protocol implementation flaws, *e.g.* using malformed packets. Asosheh and Ramezani [8] reword this classification and dig into details by citing protocols and flaws exploited in attacks.

Mirkovic and Reiher [6] eliminate the distinction of attack manifestation in the attack mechanism classification. As a consequence, a single category gathers the protocol design exploitation and implementation flaw mechanisms. Another category describes attacks that trigger large amount of traffic that might seem legitimate. Such attacks can aim at consuming bandwidth, as proposed by Douligeris and Mitrokotsa, or any other resources, *e.g.* processing capabilities, disk I/O, *etc.*

2.1.3 Characterization of the Volumetric DoS Problem

In this dissertation we focus on volumetric Denial of Service attacks. This section details the principle and characterization of such attacks. Figure A.1 shows an example network infrastructure that is hit by such an attack. The legitimate client and its traffic towards the Internet service are depicted in green while the attackers and their traffic are drawn in red. A dedicated DDoS mitigation solution protects the Internet service, for example at the datacenter entrance. The mitigation solution may implement all filtering mechanisms described in Section 2.3 to filter DDoS attack traffic.

In the case of a small attack, both the network infrastructure and the middle-box are able to cope with the volume of the attack. The middle-box filters out the attack traffic and forwards the legitimate traffic to the targeted service. When no filtering mechanism is available, both attack and legitimate traffic continue flowing towards the Internet service without causing adverse effects to equipment on the path towards the service.

A volumetric DDoS attack achieves the DoS condition by saturating the available bandwidth of a network link (*cf.*, method of attack) and causing congestion. However massive attacks can also overwhelm undersized equipment processing capabilities, *i.e.* equipment which does not meet line-rate requirements. Such equipment, *i.e.* the middle-box and/or the available resources upstream, *e.g.* router, load balancer or firewall, can be the weak point. The exhaustion is mostly due to the funneling effect that concentrates the attacks coming from distributed sources towards the target, as depicted in Figure A.1. In that case, the sum of volumes (or packets) of all traffic forwarded or routed through the same outbound interface exceeds the outbound interface bandwidth. As a consequence, exceeding packets are dropped at the bottleneck. The fact that the bottleneck is upstream from the middle-box renders filtering inefficient, as the legitimate traffic flooded among attack traffic is already lost due to congestion. The Internet service then becomes unavailable or degraded.

As a subset of traffic is still forwarded by the congested equipment through the output interface, legitimate traffic from some users may therefore partially flow towards the target. Depending on the service, their perception of the disruption can differ. This user perception of the service quality is named Quality of Experience (QoE) [9]. For example, considering a

UDP-based streaming service, packets loss can result on degraded video, long buffering time, *etc.* Conversely, other services such as those based on TCP, may even worsen the congestion. TCP integrates integrity mechanisms such as retransmission, *i.e.* a lost packet is retransmitted several times after a configured timeout, maintaining equipment under pressure by legitimate clients. This is then important to assess the impact, either partial or complete, of the attack, as discussed in Section 2.2.

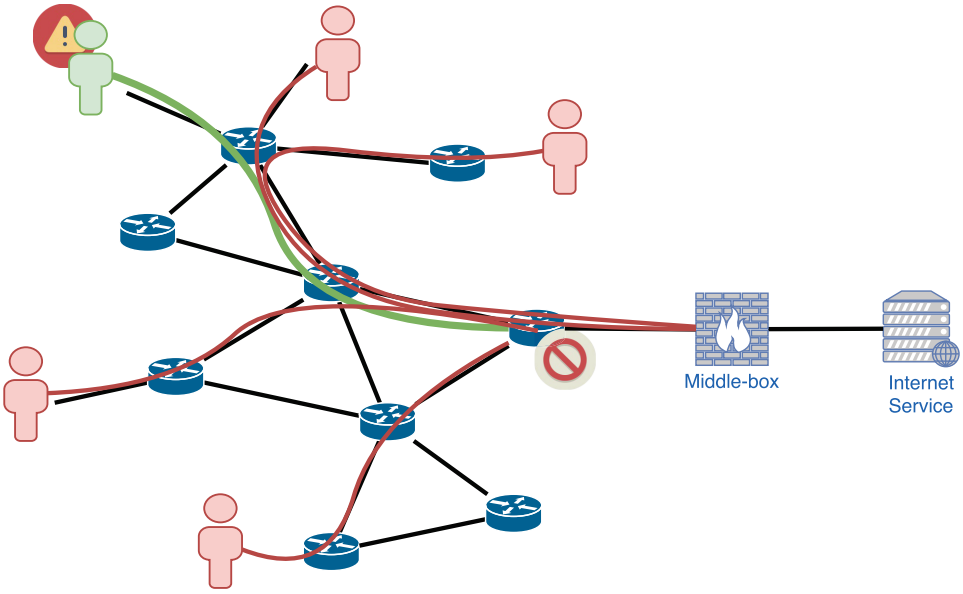


Figure 2.3 – Example of volumetric DDoS attack attack against an Internet service that saturates network link

Attack sources

As stated before, volumetric DDoS attacks leverage the distribution of attack sources to generate large amount of converging traffic. Attackers have multiple options to send traffic from such a large number of sources. Literature shows that given the source characterization, different countermeasures can be applicable.

Botnet-based attacks Botnets are groups of compromised devices (*i.e.* bots), *e.g.* servers or computers, under the control of the attacker. Devices may have been infected or voluntarily enlisted in the botnet by their owner [10]. Accordingly, the attacker is able to trigger a coordinated attack from the group of bots. Recent years have seen the emergence of Internet of Things (*IoT*)-based botnets. Due to the weak security of IoT devices such as webcams, and the large bandwidth available, IoT-botnets have rapidly

entered into the arsenal of DDoS attackers. The attack against OVH [1], holding the record of 1Tbps of accumulated attack traffic, involved more than 145,000 simultaneous IoT devices.

Bots directly send packets to the target, so that they are able to leverage attacks that require bidirectional traffic (*cf.* Table 2.1). As a consequence, from the point of view of the victim, the number of sources is constrained by the number of devices an attacker is able to control. However, the ability for an attacker to control such number of potential bots, such as in an IoT botnet, makes massive application-layer attacks (*e.g.* floods of HTTP or HTTPS POST requests containing files to upload) more likely.

However, as depicted in Table 2.1 in order to evade both detection and mitigation, bots can also send simple garbage floods, *i.e.* traffic which payload does not matter. In that case, only the volume of traffic matters and this does not require the target to reply to the real source. These bots exploit connectionless protocols, *i.e.* the protocol does not need to validate the source of traffic, such as UDP. So that they can make use of spoofing in order to evade both source detection and mitigation. In other words, they send datagrams with a faked source address. Finally, the victim may view an infinite number of sources. We do not consider such spoofed attacks in this dissertation.

Amplification DDoS More generally, Distributed Reflective DDoS attacks exploit the fact that the UDP protocol is connectionless. A bot then issues a request to a service with the source address of the UDP datagram spoofed to the target's address, resulting in the service sending its reply towards the target. If the reply is larger than the request, the server can be used as an amplifier, and the attack is called an *amplification DDoS attack*.

DNS is an example of such an exploited service, where DNS resolvers can be used as amplifiers. Many resolvers are, however, configured to respond only to a restricted set of clients, limiting the usability of the resolver as an amplifier. For example, ISPs typically allow requests only from source addresses corresponding to their customers. Resolvers without such limitations are called *open resolvers*, and they can be and are exploited for amplification attacks.

Several services are likely to be exploited for amplification. Rossow [11] studied characteristics of the use of over 14 services for amplification. He stated that a request to an NTP service can result in a reply that is 550 times more voluminous. Although he found up to billions of potential amplifiers, the attack with the highest number of amplifiers reported only around 170k different IP sources involved.

Even if the attacker initially spoofs the source address of the request, the traffic from the amplifier to the target is unspoofed. This means that the number of sources seen on an attack is limited by the number of am-

Attack	Protocol	Sources	Spoofed ?	Payload	Victims
UDP flood	UDP	Bots	Mostly	Garbage	Network
Stateful flood	GRE, TCP-based (*e.g.* HTTPS/S)	Bots	No	Compliant	Targeted service, filtering devices, network
Amplification	UDP	Internet service	No	Reply	Network

Table 2.1 – Volumetric DDoS Attack Characterization

plifiers (*e.g.* open resolvers for DNS) the attacker can find and use. In this dissertation, we position ourselves between the amplifier and the target, not between the attacker and the amplifier

Similarly, the amplification attack can be triggered by a botnet, *i.e.* bots are each issuing the request to services by spoofing the source address. However, still from a victim point of view, the fact that the attack is originated from a botnet is therefore not visible.

2.2 Assessing the Impact of DDoS Attacks

Volumetric DDoS attack aim at causing congestion on network paths towards the targeted service by saturating a link. Generally, the probability that a packet is delivered can be approximated by the ratio between the outbound bandwidth (R_O) and the sum of all incoming traffic (R_I) that is going to be forwarded through the outbound interface [12], *i.e.*:

$$P_D \approx \min(1, \frac{R_O}{R_I}) \ . \tag{2.1}$$

Probabilities have their limits, as attacks may make a huge impact regardless of probability. Researchers then have defined several metrics to assess the impact of congestion on traffic. These range from network-based metrics to more user-centric metrics and include organizational metrics. In this section, we first detail network aspects of the impact on traffic. Second, we review user-centric aspects of the impact which are mostly specific to each targeted service.

2.2.1 Network-based metrics

The use of network metrics to assess the impact of a volumetric DDoS attack, especially its congestion effect, is straightforward. The congestion basically causes packet drops at intermediate equipment.

Hachem *et al.* [13] define the Percentage of Reception (POR) metric as the ratio between the inbound and outbound traffic of a location (*e.g.* network, equipment). This metric is computed for different classes of traffic defined by the degree of legitimacy of traffic, for example legitimate traffic, suspected spams, port scanning requests, DDoS traffic. This simple metric requires the ability to classify traffic in specific categories at inbound and outbound links. While it gives a macroscopic picture of the traffic losses, it does not take into account the impact of drops at a connection level. For example for a stateful protocol such as TCP, the impact of a packet loss may be more costly. It will trigger resilience mechanisms such as retransmission and can impact the TCP transmission window size, *etc.* It may then induce delays or even session resets in case of successive packet drops for a given connection. Accordingly, considering both TCP and UDP, TCP claims more packet losses than UDP in case of congestion [14].

Zhang *et al.* [15] and other researchers [16, 17] assess the impact of congestion using the *throughput* metric. It reflects the number of packets per unit of time reaching the destination. However, considering TCP, packet losses induce packet retransmission from the sender. Each of those retransmitted packets are counted even if the information they carry does not reach the destination. Accordingly, the *goodput* metric does not include retransmitted packets. Notwithstanding, in order to give a good insight of the congestion impact, these two metrics require to be compared under the same context with and without congestion. In fact, their value is highly dependent on other conditions, *e.g.* the number of connections, the scheduling of packets on network equipments. As a consequence, their use is incompatible with a live assessment of volumetric DDoS attack impact.

Owezarski [18] study the effect of DDoS on Quality of Service (QoS). Authors evaluated the QoS using the Long Range Dependency (LRD) metric. The LRD metric aims at depicting the dependence between packets of the same connection. The dependence is due to closed control loop congestion resilience mechanisms, such as congestion window or the flow control, inherent to TCP, *i.e.* the packet delivery (rate, retransmission, *etc.*) is dictated by feedback information between the client and the server [19]. TCP congestion control mechanisms adapt the communication throughput, *e.g.* by modifying the number of successive packets the client can send before receiving an acknowledgment (congestion window), to fit the link bandwidth, so that it traffic throughput oscillates. However, when the range of oscillations increases, *e.g.* due to a long-term dependency, the oscillation amplitude also increases. This induces TCP connections throughput to reach even lower minima.

As evoked in this section, simple network metrics lack the understanding

of the type of traffic, *e.g.* a packet loss for a given protocol does not lead to the same results for another one. The typical low-level example is TCP and UDP. While a packet loss for TCP induce throughput reduction to adapt the connection to the link bandwidth, it does not have such effect for UDP traffic. To be precise, UDP traffic may adapt to bandwidth constraints, *e.g.* streaming traffic. However, this mechanism is not implemented at the UDP layer, but this is an application-level mechanism.

2.2.2 User centric metrics

An attacker does not need to saturate a network to disrupt a service. He only needs to send a sufficient amount of traffic to degrade the availability of the service. As a consequence, the legitimate user will have a very poor experience of the service. Conversely, a user may not notice a degradation of its experience even if a few networks packets have been dropped [20]. In fact, the quantification of a poor user experience depends on the type of application and especially on the tolerance of the application to network losses.

The International Telecommunication Union (ITU), for example [21] provides end-user oriented recommendations for multimedia QoS metrics. They define application categories shown in Figure 2.4, for which services are classified by type and QoS requirements. Requirements are assessed using three QoS metrics that have been identified as reflecting the impact on user perception, namely QoS delay, jitter and information loss. For example, a delay undoubtedly impacts the immediacy and interactivity of a telnet session, while a high jitter degrades the quality of a VoIP conversation. Finally, a loss of packets leads to frame loss and / or inconsistent buffering for streaming related traffic. However, depending on the class of traffic they are applied to, these metrics have different meanings. For instance, delay either refers to the time between the arrival of two consecutive packets or the duration of a download. Similarly, these metrics are not clearly defined. For example, according to the ITU report, HTTP traffic does not tolerate losses. A loss is surely identified by the timeout of a TCP session, and induced by the loss of a packet and its re-transmission.

Mirkovic *et al.* [22] propose a set of metrics to measure the degradation of quality of service and thus the severity of a DDoS attack. Their work is based on a unified QoS metric for all different classes of traffic, *i.e.* the failure of an application transaction. The transaction, *i.e.* the core of their QoS evaluation, depicts an application-level task. Then, similarly to ITU, the QoS failure is evaluated using service tolerance to traffic errors, namely one-way delay, request-response delay, loss, duration and jitter. Although the method to capture these metrics also change regarding the type of traffic,

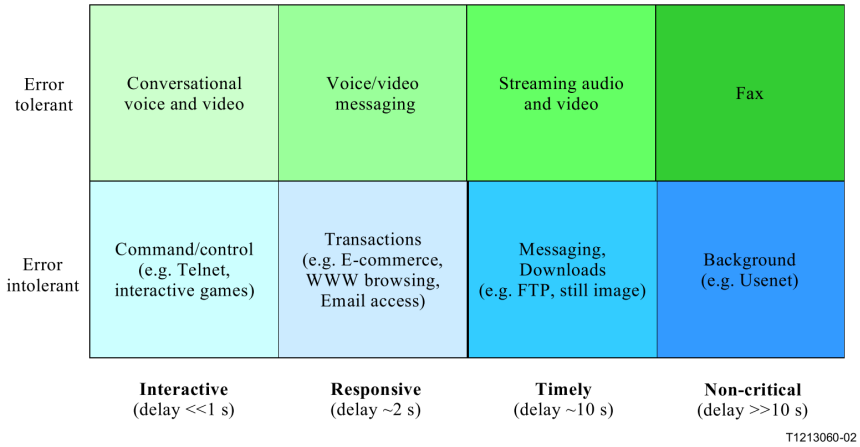


Figure 2.4 – ITU User-centric QoS recommendations

Mirkovic clearly states how to measure them. Finally, the severity of the attack is expressed using a combination of percentages of failed transactions per application.

2.3 Admission Control for DDoS Mitigation

Admission control describes mechanisms that decide if a service request should pass the mechanism or be blocked. The request can be identified at different OSI layer, *e.g.* layer 3 to layer 7, so that the decision is made on information contained in the fields of these layers.

This section presents a state of the art of admission control mechanisms that can be applied to mitigate DDoS attacks. Subsection 2.3.1 presents active mechanisms, *i.e.* that require modification of the connection traffic and monitor the behavior of the client to make their decision. Subsection 2.3.2 then presents passive mechanisms that do not require modification of traffic.

2.3.1 Active Admission Control

We split research for active admission control into three categories. The first category aims at identifying suspicious clients by looking at their behaviors when facing a challenge, *i.e.* a modification of the traffic. The second category uses client capabilities, *i.e.* authorization granted by a server to end-hosts. The last category is based on marks applied to packets along the path to the server, that are checked by routers.

Challenge-based mechanisms

Puzzle-based challenges Puzzles are challenges sent by a service to clients, that are moderately hard to solve, but simple to check. The basic idea of puzzles is that clients should pay a computational price, *i.e.* higher than just sending a request, so that an attacker would require more computational resources to carry out the attack. These challenges are mostly based on a brute-force search of an entire cryptographic pre-image given a partial pre-image and the image.

Juels and Brainard [23] first introduce the concept of puzzles to mitigate DDoS, and especially connection depletion attacks such as SYN floods. Such attacks aim at consuming the connection backlog of a server by multiplying connection attempts. In normal operation, clients ask for puzzles, then servers register them without sending them a challenge. During attacks, the servers reply to clients with a puzzle and only registers them after checking that the solution they replied is valid. Puzzles are time-based and stateless, so that, clients have limited time to solve the challenge and servers do not have to keep tracks of generated puzzles to verify the solutions. The size of challenges is computed in such a manner that it is theoretically impossible for an attacker that control a given number of bots to achieve a connection-depletion attack. As a consequence, in 1999, the puzzle is recommended to be a 128 Bytes bitstring. This means that today, according to Moore's law, the puzzle's size should be much larger, leading to an extra burden of traffic bandwidth. Moreover, the use of a puzzle protocol requires its implementation in client machines. Although the authors argue that this could be built into browsers, *e.g.* as a plugin, such implementation would not be useful to fight against SYN flood attack as the TCP connection is established into the kernel before the information is transmitted to the browser.

In 2004, Wang and Reiter [24] proposed the use of puzzles to mitigate volumetric DDoS attack. During an attack, congested routers require clients to regularly solve puzzles with a rate commensurate with the bandwidth consumed by the client. A collaborative mechanism allows congested routers to require the help of downstream routers to deliver puzzles and check their solution. More recently, in 2011, Khor and Nakao [25] proposed the use of puzzle mechanisms as a cloud-based DDoS mitigation service. They focused on billing protected services for only legitimate traffic.

TCP challenges TCP challenges act as authentication methods that check the legitimacy of the source host. In particular, they aim at tackling spoofed attacks.

For example, TCP SYN proxies [26, 27] are middle-boxes, through which all the traffic is going. In such a manner, they complete the TCP handshake in place of the TCP server. If the source is able to set up a connection, the proxy resets the initial connection and whitelists the source. Later connec-

Verifies	HTTP Redirect	HTTP Cookie	Javascript challenge	CAPTCHA
Non-spoofed Source IP	X	X	X	X
HTTP Compliant Application	X	X	X	X
Real Browser			X	X
Real Human				X

Table 2.2 – Application Challenges [27]

tion attempts are directly forwarded towards the server. Other challenges have been proposed [27]. For example, instead of trying to complete the handshake, the proxy replies with an out of sequence SYN-ACK and waits for the reset from the client. Thus, the proxy avoids the burden of sending the reset, which could dramatically increase during an attack. Nevertheless, a SYN proxy is commonly implemented in off-the-shelf mitigation mechanisms. These solutions shift the burden of establishing a first connection with the sources to the proxy. As a consequence, during an attack, the proxy may itself be vulnerable to attacks such as SYN floods.

Application challenges Application challenges mitigate applicative floods such as HTTP floods, HTTP floods consists in sending HTTP requests with a high rate to a server. Application challenges may be categorized into four sections, namely HTTP redirection, HTTP cookie, JavaScript challenges and CAPTCHA. As shown in Table 2.2, each mechanism checks different elements [27].

These challenges may be implemented on the victim side of a DoS [1]. Nonetheless, this can also be embedded in middle-boxes together with a set of dedicated mitigations.

An HTTP redirection consists in replying to a client request with a 302 HTTP code, that redirects the client to a link that can also be intercepted by the proxy. Once the proxy detects that the client has followed the redirection, the source (or connection) is whitelisted. An HTTP cookie follows

[1] TestCookie Nginx module http://kyprizel.github.io/testcookie-nginx-module/

the same principles and adds a cookie in the redirection announcement. The client then needs to follow the redirection with the embedded cookie to be authorized to fulfill the request. However, even basic downloading tools, such as the UNIX command wget [2] are able to successfully bypass these protections.

Javascript challenges go further and try to check that the environment of the client is a browser. Simple JavaScript challenges, *e.g.*, mathematical calculations, are embedded into a dedicated HTTP content. Completing the challenge requires the client application to interpret and run the JavaScript code. Such mitigation is circumvented by some sophisticated botnets, that integrate a JavaScript interpreter.

CAPTCHA are tests that aim to differentiate humans from computers. The client has to answer a visual test, for example, to reproduce a text shown in a degraded image or to achieve a calculation depicted in a picture. Even if they have been designed to be costly to solve, they can be deceived using for example Optical Character Recognition, Artificial Intelligence, *etc.* as proposed in [28].

Capability-based mechanisms

Capabilities are tokens, carried by packets, which represents the entitlement of a client to send traffic. This authorization is granted by the service, which decides which client to serve. Although the principle is not restricted to client-server architectures, in this section, we respectively denote as client and server the initial sender and its receiver.

Yaar *et al.* [29] propose the Stateless Internet Flow Filter as a countermeasure to flooding DDoS attacks. It is based on the concept of privileged and unprivileged traffic, *i.e.* traffic which respectively carries valid stateless capabilities and no capability at all (legacy traffic). Capabilities are generated using the client IP address and marks generated by routers and delivered by the server if it decides to authorize the client. The server is then able to deny the capability for suspicious clients. Privileged traffic will be prioritized at routers after that the capability extracted from a modified IP header of the packet has been validated. Routers validate a capability by checking that the router mark is present. Packets with an invalid capability are dropped, while packets with no marks, are de-prioritized.

The use of capabilities allows mitigating floods of unprivileged traffic. As a consequence, the resilience of the mechanism is based on the fact that the server is able to detect malicious sources to deny their capabilities. However, in most cases, the service will classify a source as malicious after the source's

[2] Wget tool https://www.gnu.org/software/wget/

request hit the service. In other words, attacking sources are able to send at least a request before being identified as malicious.

Yand *et al.* [30] aim at solving the vulnerability of previous capabilities mechanisms during the time they grant or deny access to a client, as they require reading at least the first request to grant or not the capability. They introduce the Traffic Validation Architecture which is also based on capabilities checked by routers. They prevent servers from being vulnerable to flooding attacks during the capability granting phase by limiting requests per flows. Flows are identified at a router by the path taken by the packet towards the router, and the packet destination. Routers thus store a small flow state. However, this solution still requires an implementation of the capability protocol on end-hosts and routers. Moreover, the mapping of an identifier to flows is vulnerable to floods. In fact, legitimate and attack flows towards a given destination which hit the protected network at the same access point would get the same path identifier. As a consequence, they would be rate-limited jointly, and legitimate traffic is likely to be shadowed amongst the attack traffic.

Discussion

Challenge mechanisms focus on the idea that a client has to provide something, either proof of its legitimacy or a payment, *e.g.*, in the form of CPU usage. This, however, requires an additional step in service delivery. Indeed, the service replies to a client request with a challenge. It then waits for the challenge completion to answer the initial service request. This principle is not applicable to all services, especially those who require high availability. For example, UDP-based DNS service relies on the spontaneity of the stateless communication between the client and the server. As many other Internet services require domain name resolution, an increase of the latency in DNS resolution has a dramatic impact.

Capability-based mechanisms assume that the server can discriminate attack from legitimate traffic and grant capabilities to the legitimate traffic. However, especially in the case of volumetric attacks, choke point occurs upstream from the destination, so that legitimate traffic is already flooded amongst the attack traffic. Newer proposition provide enhancement by rate-limiting undetermined traffic but push the congestion at the network access point.

2.3.2 Passive Admission Control

Passive Admission control mechanisms broadly fall into three categories: filtering mechanisms, queuing-based congestion control mechanisms and combination of the two previous categories. Filtering focuses on matching traffic

that has been identified as unwanted according to either a pattern or a behavior. Conversely queuing-based congestion control mechanisms aims at using network resources as efficiently as possible in order to avoid congestion collapse [31]. Congestion collapse is a critical network state where packets are retransmitted (*e.g.* related to TCP traffic) several times before initial transmission leaves the network, *i.e.* is being effectively dropped or received by the destination [32]. This is the case in networks that first delay packets when facing link saturation to comply with bandwidth limitations, resulting in a rise of the Round Trip Time, *i.e.* the time for a packet to be sent to the destination and for the acknowledgment to be received by the sender. The Network contains several copies of the same packet, which make the matter worse.

Filtering mechanisms

Filtering mechanisms map a forwarding decision, *i.e.* either pass or drop, to a part of traffic matched by a pattern or a behavior. More complex actions can also be assigned to a traffic profile.

The use of existing and already deployed network equipment to achieve a first line of defense has been proposed by the industry. Blackholing, such as described by Cisco [33], provides a simple and resource-efficient method [34] to drop a collection of packets based on their destination or source prefix. Indeed, it uses the efficient routing mechanisms of routers to route packets, destined to a given destination IP address or prefix, to a null route. Traffic is then purely discarded. A destination-based blackhole would, however, disrupt the service by entirely dropping the traffic routed towards it.

Access Control Lists (ACLs) are filters typically implemented in network equipment [35]. They consist of an ordered list of rules that define a part of the traffic using packets fields, such as IP addresses or TCP/UDP ports, and an action that is assigned to the matched traffic. Rules are sequentially tested against the traffic until a match happens. If traffic does not match a rule, it is passed to the next forwarding step. Fields are often restricted to OSI Layers 3 and 4 [35]. The most prevalent actions are DENY or ACCEPT the packet that is matched by the rule, *i.e.* the packet is respectively blocked or passed to the next forwarding/routing step. Whitelists are a specific type of ACLs where the only available action is ACCEPT. Its last rule catches all the traffic and denies it. Filtering traffic with whitelists requires the identification and characterization of wanted traffic (*e.g.* legitimate traffic, or protected service). Conversely, blacklists only allow the DENY action in rules and accepts the remaining unmatched traffic. In that case, the unwanted traffic such as malicious traffic, has to be characterized.

Today, ACLs are often implemented in hardware [36, 37], for a reduced performance impact, instead of in the router software. On the other hand,

hardware implementation comes with size constraints [38]. Filtering lists are stored in a fast but expensive memory (TCAM) which is size-limited [39]. Maccari *et al.* [40] propose the use of a memory efficient structure (Bloom Filter [41]) to reduce the size occupied by a whitelist. [42, 36, 39, 43] considered the memory limitation and aimed at reducing the size of lists using source prefix aggregation. Several aggregation schemes have been proposed in the literature.

Network Aware Clusters [44] identify topologically-closed sources using the BGP routing table. Authors assume that BGP route prefixes identify nodes (IP addresses) in the networks which are topologically close. The more those prefixes are long, the closest the nodes.

The Hierarchical Heavy Hitters [45] extends the heavy hitters algorithm, which produces clusters which rate (throughput, bandwidth, *etc.*) is greater than a configured threshold, and takes into account the hierarchical structure of IP prefixes. The Hierarchical Heavy Hitters algorithm produces aggregates with approximately equal rates of legitimate traffic.

Pack *et al.* [39] study the ability of filtering lists (whitelists, blacklists and a combination of both) to filter malicious traffic while preserving legitimate traffic. Aggregates are computed using a comparison between a baseline period of traffic and the last period of traffic.

Goldstein *et al.* [42] also propose a history-based algorithm to automatically generate filtering rules using Bayesian decision theory. Authors aim at constraining the number of rules. They achieve Bayesian classification with a given network prefix size. Adjacent produced clusters (*e.g.* 1.1.2.0/24 and 1.1.3.0/24) are aggregated. If clusters are still too numerous, classification is recalculated with a shorter prefix.

Soldo *et al.* [36] develop a framework to build optimal ACLs, where the definition of an optimum depends on the filtering goal, *e.g.* blocking all sources, some sources, preserving bandwidth, *etc.* The aggregate computation is driven by weights assigned to each source. Although, they evoked a different method to assign weights to sources, the importance of these weights has not been assessed. In Chapter 4, we evaluate the impact of two weight assignment methods driven by the amount of information about traffic.

Dedicated filtering middle-boxes Network appliances, also known as middle-boxes, are widely used to mitigate DDoS attacks [5]. This generic term depicts various appliance types such as network and application firewalls, IPS, proxies or dedicated appliances (*i.e.*, anti-DDoS). Dedicated solutions, mostly commercial solutions (*cf.* those proposed by Arbor Networks[3],

[3]Arbor Networks https://www.netscout.com/arbor

Radware[4], 6cure[5]), achieve better (*i.e.* fine-grained) filtering of malicious traffic using specifically crafted detection and mitigation mechanisms compared to other kinds of middle-boxes. Moreover, dedicated solutions are specially designed to handle a high rate and volume of traffic and applicative floods, where traditional defenses are overwhelmed.

With the growth of core network resources, and especially link bandwidth, appliances have to process traffic for deep packet inspection (DPI) at line rates that reach up to 100Gbps [46]. Despite the performance gains that follow the Moore's Law, single-threaded architectures failed to scale to the packet rate processing demand [47]. Literature, thus, first, promotes the offloading of packet processing to Graphics Processing Unit (GPU) that allow massive parallel computation [48, 49, 50, 51] and then Network Processing Unit (NPU), *i.e.*, network cards with hardware processing capabilities such as Field-Programmable Gate Array (FPGA) [52, 47]. IDSs and IPSs can fully benefit from the parallelizing of packet processing, even when dealing with deep packet inspection, as they consist of matching traffic against a large number of rules [53]. Conversely, statistical or behavioral detection and mitigation of DDoS is less suitable to parallelization. Indeed, flow behaviors are detected using various information [54, 55], requiring a pipeline-oriented processing of packets, as opposed to parallel processing [53].

Trevisan *et al.* [53] proposed a software architecture with parameter tweaks to achieve statistical traffic analysis at 40Gbps. They use the Intel Data Plane Development Kit (DPDK) [6], a software framework that provides fast packet processing by directly accessing hardware from userland, suppressing the overhead induced by the call on kernel functions.

Tokusashi *et al.* [56] proposed an FPGA-based DNS amplification DDoS attack mitigation appliance. The middle-box tracks DNS responses and adds the source and destination IP addresses and ports 4-tuple to a hash table. The appliance also monitors ICMP port unreachable messages in order to check if it has been caused by a DNS response. If so, the related 4-tuple is blacklisted. The simplicity of the mechanism allows parallelization of packet processing, as it mainly consists of matching packets against hash-table.

Jyothi *et al.* [57] proposed to offload parallelizable tasks to the FPGA, especially the packet parsing up to the application layer of text-based protocols (HTTP, SIP, *etc.*). The parsing of such protocols is computationally expensive as it requires the search of header fields' name in the application layer data.

Benacek *et al.* [58] follow the same idea by providing an automated means to generate protocol stack parsers, based on the P4 language [59]. The P4 language is used to define how protocols are stacked and then describe the

[4]Radware https://www.radware.com/
[5]6cure https://www.6cure.com
[6]Intel DPDK https://www.dpdk.org/

workflow to parse packet data.

One of the major drawbacks of dedicated middle-boxes is their price, which increases with the bandwidth of links to protect. As a consequence, the distribution of such appliances inside the network or at its access point is hampered by the cost.

Hop Count Filtering [60] relies on the Time To Live (TTL) field of IP packets to determine the legitimacy of traffic. The TTL field aimed at storing the number of hops (*i.e.*, gateways) crossed by the packet since its generation. Hop Count Filtering works on the premise that, at a given inspection point in the Internet, the TTL field of packets from a same source have substantially the same value, despite the fact that packets may travel down different paths. By detecting inconsistency in packet TTL fields, the filtering mechanism can discard packets with spoofed source IP addresses. The drawback of such mechanism is that the attacker is also able to spoof the TTL field, so that it seems consistent. In fact, an attacker can learn, for example from global BGP maps, the Internet topology and guess the correct TTL values of related spoofed source IP addresses. Moreover, [61] argues that during congestion, for example caused by an attack, the traffic path can consequently be modified to prevent losses, resulting in an alteration of legitimate traffic hop counts and in an increase of the risk of false positives and negatives.

Scheduling mechanisms

Queues are important data structures used in network equipment to organize and schedule packet departure according to deterministic or probabilistic parameters. In network packet switching/forwarding, scheduling refers to methods that define how to choose which packet is going to be first handled (routed, sent, ...) in queues. Table 2.3 shows a review of the scheduling-based congestion mechanisms.

Class-based Queuing (CBQ) [62] is based on the notion of Class of Traffic (CoT), which are mutually exclusive subset of traffic. Classes of Traffic can define, for example, applications, protocol families, customers, ... CBQ provides schedulers based on a link-sharing structure. It allows allocating bandwidth to classes of traffic, such that each CoT may be restricted to their link-sharing bandwidth. In order to avoid link under-utilization when some class does no use its allocated bandwidth, their remaining (*i.e.*, unused) bandwidth may be shared among the other classes that exceed their own allocated bandwidth. While CBQ allows guaranteeing fairness between classes, it fails to mitigate volumetric attacks when both legitimate and attack traffics belong to the same CoT.

Fair Queuing (FQ) [63] aims at enforcing fairness between flows by allocating a queue for each couple of source-destination IPs. The scheduler

Mechanism	Allows per-flow treatment?	Flow granularity	Require flow state storing?	Guarantee fair share?	Prioritized flow characterization	Deprioritized flow characterization	Require Protocol Change?	Robust to UDP-based DDoS?
CBQ 1995[62]	No (same treatment per class)	Class	No	No		High bandwidth flows	No	Yes (but does not protect legitimate UDP flows)
FQ 1989[63]	Yes (1 queue per flow)	src IP?	No	Yes	intermittent Telnet	FTP	No	Yes?
SFQ 1990[64]	Yes	src IP, dst IP	No	Yes	Telnet	FTP	No	Yes (small number of flows)
CSFQ 1998[65]	Yes	N/C	Yes (edge routers)	Yes	fair or responsive flow (eg. TCP)	Unresponsive/unfair (eg. UDP)	Yes (collaboration with core routers)	Yes
DRR 1996[66]	Yes	src IP, dst IP	No	Yes	Telnet	FTP (bandwidth consuming)	No	Yes

N/C: Not Communicated
?: Not communicated, guessed from the bibliography

Table 2.3 – Synthesis of scheduling-based congestion control mechanisms

selects packets to dequeue so that equal bandwidth is guaranteed between all active queues, *i.e.*, non-empty queues. FQ requires equipment to allocate and maintain a queue for each source-destination pair, which is not scalable as it consumes a lot of memory. Stochastic Fairness Queuing (SFQ) [64] and Deficit Round Robin (DRR) [66] focus on this issue and propose to distribute flows to queues according to their addresses hash. As a consequence, the number of queues to allocate and maintain is limited by the hash domain. Results show that fairness between flows increases with the number of allocated queues, as that flows whose hash collide are not treated fairly with each other.

Core Stateless Fair Queuing (CSFQ) [65] is built upon the differentiation between edge and core routers. Edge routers handle fewer flows, thus may perform more complex tasks than core routers. As a consequence, edge routers perform flow tracking and labeling, so that core routers only process packets using the label and local measurements. However, [67] argues that CSFQ is unable to ensure the fairness of flows, especially in case of large and bursty flows

Queuing-based congestion control management

Starting from the global network congestion problem, *i.e.* whether saturation is caused by a malicious behavior or not, researchers have extended existing solutions and proposed new ones to deal with volumetric DDoS attacks by managing the content of queues. They mainly aim at handling traffic bursts and over subscription, therefore avoiding avertible losses by protecting traffic from the effect of other harmful traffic[7]. Table 2.4 provides a summary of review of queuing-based congestion control mechanisms.

Random Early Detection-based queue management The major drawback of queues is that they induce latency and jitter. They grow with the queue length, *i.e.* the number of packets in the queue.

Floyd *et al.* [68] propose the Random Early Detection (*RED*) mechanism, a probabilistic algorithm that drops (or marks) packets before they are transmitted to queues. The probability of a packet being dropped is a function of the average number of packets in the queue. Two bounds are configured, so that when the average queue length is between these two values, the packet drop probability linearly grows from 0 to 1. RED then reduces delays and jitter by bounding the queue length. This prevents packet retransmission due to timeout mechanisms so that the retransmitted packet hits the network before the original packet leaves it, resulting in worsening congestion. However, RED is not able to prevent high loss rates in case of congestion, let alone preserve legitimate traffic during a volumetric DDoS

[7] https://datatracker.ietf.org/wg/aqm/about/

Mechanism	Allows per-flow treatment?	Flow granularity	Require flow state storing?	Guarantee fair share?	Prioritized flow characterization	Deprioritized flow characterization	Require Protocol Change?	Robust to UDP-based DDoS?
RED 1993[68]	No	N/A	N/A	No	N/A	N/A	No	No
ARED 2001[69]	No	N/A	N/A	No	N/A	N/A	No	No
DRED 2001[70]	No	N/A	N/A	No	N/A	N/A	No	No
FRED 1997[71]	Yes	src IP, dst IP, src port, dst port, protocol ID	flows in queue	No	fragile and robust responsive flows	unresponsive flows	No	Yes?
SRED 1999[72]	some high bandwidth flows (random selection)	src IP, dst IP, src port, dst port, protocol ID	High-bandwidth	No	N/A	N/A	No	No
RED-PD 2001[73]	some high bandwidth flows (random selection)	src IP, dst IP, src port, dst port, protocol ID	High-Bandwidth	No	responsive flows	high-bandwidth unresponsive flows	No	Yes?
StoRED 2007[74]	Yes	N/C	aggregated packet count (counting Bloom filter)	No	responsive flows	unresponsive UDP flows, parallel TCP downloads (multiple connections, same flow)	No	Yes (for a reasonable number of flows)
CHOKe 2000[75]	some high bandwidth flows (random selection)	N/C	No	No		unresponsive and unfriendly flows	No	Yes?
xCHOKe 2002[76]	some high bandwidth flows (random selection)	N/C	Yes	No	TCP	UDP	No	Yes?
RECHOKe 2007[77]	some high bandwidth flows (random selection)	N/C	Yes	No	TCP	UDP	No	Yes?
PUNSI 2006[78]	some high bandwidth UDP flows	N/C	No	No	TCP traffic (flood or not)	only UDP-based floods	No	Yes?
BLUE 2002[79]	No	N/A	No	No	N/A	N/a	No	No
SFB 2001[80]	Yes	N/C	kind of high bandwidth flows storage (via Bloom filters)	No	TCP responsive flows	TCP unresponsive	No	Yes (for a reasonable number of flows)
RSFB 2009[15]	Yes	N/C	Yes	No	responsive FTP traffic	spoofed UDP floods	No	Yes (even spoofed DoS)
Drop-Sel 2011[81]	No (per class treatment)	N/A	N/A	No		N/A	No	Yes?
DFS 2015[16]	Yes	src IP, dst IP	Yes	Yes	TCP responsive flows	UDP floods	No	Yes

N/A: Not Applicable, mostly because the algorithm does not aim at prioritizing flows over the rest. This is the case for example for RED and some of its derived works, that only reduce the queue loss. On the other case this may be due to the fact that algorithm aims at controlling congestion regardless the origin, *i.e.* whether this is caused by a flood or not.

N/C: Not Communicated
?: Not communicated, guessed from the bibliography

Table 2.4 – Synthesis of review on queuing-based congestion control mechanisms

attack.

Multiple works have focused on the aforementioned RED issues. They start by identifying either unfair flows, *i.e.* that consume more bandwidth than a fair usage, or unresponsive flows, *i.e.* that do not adapt their rate after being notified of congestion. Packets belonging to these flows are de-prioritized, *e.g.* simply dropped, or have a higher drop probability assigned.

Flow RED (FRED) [71] achieves fairness between flows by applying RED to each flow, *i.e.* it maintains the number of packets for each flow in the queue. RED parameters are then fixed per flow instead of for the whole traffic.

Stabilized Random Early Drop (SRED) [72] also stores per-flow state but for all flows. To avoid keeping large tables of flow states, timeouts are applied to flow information.

RED with Preferential Dropping (RED-PD) [73] identifies flows sending more than a given rate, using the history of RED packets drops.

Stochastic RED (StoRED) [74] hashes flows into bins and increments matching bins. The probability of drop depends on the matching bin counter. While this is efficient for a very small number of flooding flows, StoRED is inefficient for a large number of flows as in volumetric DDoS attack.

CHOKe-based algorithms Also based on the RED approach, CHOKe [75] compares the arriving packet to a randomly chosen one in the queue. If the average queue is above the minimum RED threshold. If both packets belong to the same flow, both packets are dropped. Otherwise, it acts like RED algorithm. Experiments show that CHOKe sometimes penalizes fair flows (*e.g.* TCP flows) by successively dropping packets from the same flow. Moreover, while CHOKe does not maintain flow states, its performance is limited under a volumetric DDoS attack, *i.e.* high-bandwidth flows are not tied to a fair bandwidth, as even these flows have a few packets in the queue at a given router. So that probability of finding packets belonging to the same flow is low.

xCHOKe [76] aims at solving the last issue by maintaining a list of flows for which two packets have matched. To avoid a long list, flows expire after a fixed time. The CHOKE algorithm is then pre-filtered with this list. RECHOKe [77] combines both xCHOKE and RED-PD approaches in order to deprioritize faster than CHOKE and xCHOKE.

PUNSI [78] solves the first CHOKe issue by adding dedicated treatments to UDP packets. TCP traffic drop probability remains computed with RED algorithm. This is intended to be more robust against UDP flood-based DDoS attacks. However, UDP traffic is heavily de-prioritized compared to TCP traffic [81].

RED derivatives work by forcing flows to a fair rate. Traffic based upon protocol with congestion management mechanisms such as TCP adapt their

rate so that it will be less penalized by the queue management. On the contrary, protocols such as UDP, without a sending rate adaption mechanism implemented at the application level, are intended to be constrained to this fair share rate. Such approach only works for volumetric DDoS attacks targeting TCP-based services.

BLUE-based mechanisms BLUE [82] does not rely on the queue length to manage congestion like RED and its derivatives. Instead, it is based on packet losses and link idling to respectively increase or decrease a drop probability. This approach lacks fairness enforcement between flows, so that authors also proposed Stochastic Fair Blue (SFB) [80]. Like StoRED, SFB hashes flows into bins. Then the BLUE algorithm is applied for each bin, *i.e.* it maintains a per-bin drop probability. But similar to StoRED, TCP flows suffer from high drop rates when the traffic contains a large number of high-bandwidth flows, *e.g.* spoofed flows. Resilient SFB (RSFB) [15] has been thus proposed as an alternative robust to spoofed DDoS attacks. It maintains a list of flows that adapt their rate as a response to packet drops.

Other anti-DDoS oriented queuing mechanisms Since different traffic have different quality of service requirements (*cf.* Subsection 2.2.2), Drop-Sel [81] proposes a traffic class aware packet drop selection algorithm. It broadly considers the following three traffic classes: real-time and non real-time UDP classes and TCP class. Drop-Sel maintains the count of packets in the queue for each traffic class. The treatment for the class with the highest occupancy is applied each time a packet arrives.

Deterministic Fair Sharing (DFS) [16] aims at providing a per-flow treatment. IT stores for each flow the number of packets in the queue. The fair flow queue occupancy is computed at each packet arrival and the packet drop probability is computed in function of the flow queue occupancy. However, as it only uses the count of packets in queue, DFS like CHOKe may not work properly when very few packets of the same flow are at the same time in the queue, as it will not detect those flows as aggressive.

Policing and Shaping mechanisms

Policing and shaping mechanisms aim at guaranteeing that traffic does not exceed a configured bandwidth. Unlike scheduling mechanisms, they do not aim at guaranteeing fair share between subsets of traffic, although they may form the basis for schedulers.

While policing mechanisms drop noncomplying traffic, shaping mechanisms smooth traffic over time and drop it only when it is impossible, as shown in Figure 2.5. As a consequence, policing induces more losses than shaping, but shaping may increase packet delays [83].

Token and leaky buckets are two commonly used rate-limit algorithms in off-the-shelf equipment [84]. Both require the configuration of a committed average rate, *i.e.* the long run bandwidth limit of the mechanism. They may also allow short-term traffic bursts within the configured limits.

Distributed mechanisms

In order to prevent congestion at a node caused by the attack traffic convergence, some researchers proposed solutions to distribute admission control upstream from the bottleneck. Table 2.5 provides a simple review of such mechanisms. The section below details these works.

Aggregate-based Congestion Control (ACC) [85] rate limits destination (or source) prefixes that have been identified as causing congestion. RED packet drops are used to identify high-bandwidth aggregates. A pushback mechanism is also proposed by the author to deploy filters upstream from the router. While destination-based rate limit can prevent congestion at the choke point, it would impact legitimate traffic by also rate-limiting the legitimate flows towards targets, which are flooded among the attack traffic. In amplification DDoS, rate limiting the source, *i.e.* amplifier IP, may result in a limitation of the service traffic, regardless the legitimacy of the request. As a consequence, filtering traffic only with the help or source or destination IP addresses may result in huge amount of collateral damages. Moreover, detecting aggregates requires maintaining numerous states (aggregate monitoring) at a router. The pushback-based mechanism has also been proposed by other researchers [86, 87, 88] to help congested routers that suffer from the funneling effect of the attack so that they cannot handle all incoming traffic. Indeed, they distribute filtering and rate-limiting order upstream from the choke point. The difference with Backward Traffic Throttling (BTT) [88] is that it does not rely on a router-based identification of attack flows, but rate-limit all traffic routed through the congested router. As a consequence, it does not require implementation of new data plane mechanisms and eases the cooperation between ASs.

BGP configurations The Border Gateway Protocol (BGP) is a widely used protocol to exchange routing information. BGP advertises that a network prefix can be routed through the AS responsible of the advertisement. A simple example is the blackholing configuration, where the advertised next-hop of a prefix is a null route, *i.e.* the blackhole. As such, traffic routed through the null route is automatically discarded. This forms a powerful destination-based blacklist, as it combines the proven efficiency of table lookups and the simplicity of traffic discarding.

Remotely Triggered Black Hole (RTBH) [33] consists of distributing the null route to the edge of a network with the help of BGP advertisements.

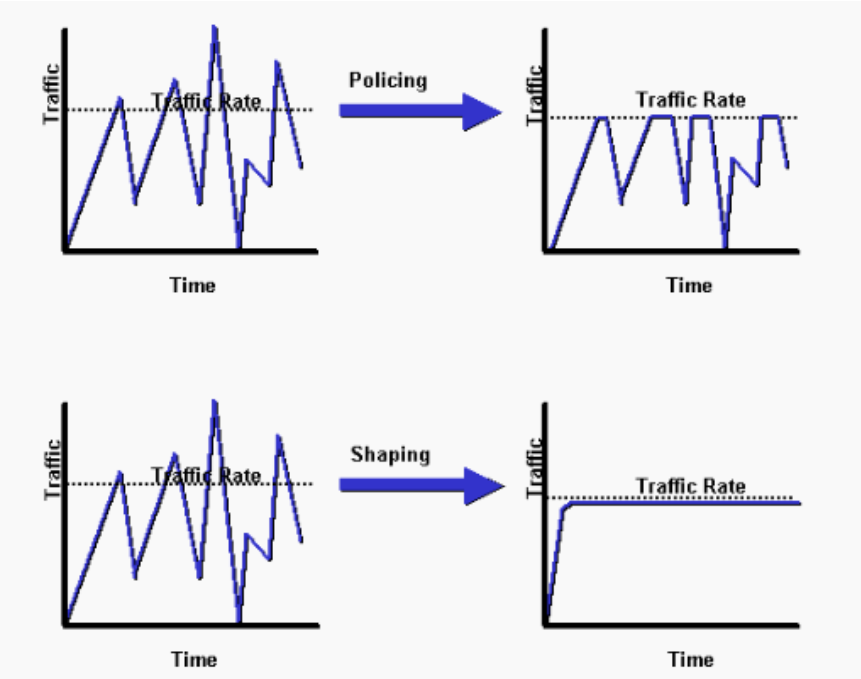


Figure 2.5 – Policing versus shaping mechanisms [83]

Mechanism	Allows per-flow treatment?	Flow granularity	Require flow state storing?	Guarantee fair share?	Prioritized flow characterization	Deprioritized flow characterization	Require Protocol Change?	Robust to UDP-based DDoS?
Distributed mechanisms								
ACC 2002[85]	Yes	either src IP or dst IP	Yes	Yes		high-bandwidth aggregates (src IP or dst IP)	Yes (push-back messages)	Yes (using collaboration with other routers)
[86]								
[87]								
[88]								
BTT [88]	Yes	all fields that identifies traffic forwarded through a downstream link	AS monitoring	No	N/A	N/A	Congestion Notification	prevent congestion
Destination-based BH [33, 89]	Yes	dst IP	No	No	N/A	N/A	No	prevent congestion
Source-based BH [33]	Yes	src IP	No	No	N/A	N/A	No	prevent congestion
[90]								
[91]								
[92]								
Combined mechanisms								
D-WARD 2002[93]	Yes	src IP, dst IP	Yes	Yes		TCP, ICMP, UDP floods	may require TCP congestion changes	Yes
Transport-aware IP router [94]	Yes	dst IP, transport protocol, TCP flags, ...	No	No	control channel	data channel	No	Yes
IDR [95]	Yes (hash)	dst IP	Yes (Counting Bloom Filter)	No	small bandwidth flows	high bandwidth flows	No	Yes (for a reasonable number of flows)

Table 2.5 – Synthesis of Distributed and Combined mechanisms

Source-based RTBH provides the ability to filter traffic based on the source IP address. It combines BGP route distribution with the Unicast Reverse Path Forwarding [96] mechanism. A null route is installed into the routing table of the external interface of the edge router by a BGP announcement. When validating the source IP address of inbound traffic, the URPF mechanism will fail with the blackholed source IPs, resulting in discarding the traffic. Source-based RTBH prevents blackholing an entire destination prefix, which causes a lot of collateral damages. The source prefix is filtered at each edge router, regardless of the destination status (target or legitimate). No details are provided on the scalability of the mechanism, *i.e.*, how many source prefixes can be filtered at a router. RTBH is typically deployed inside an AS [33], using internal BGP (iBGP).

Selective Blackholing [89] reduces collateral damage of destination-based remote triggered blackholing by carefully selecting the edge routers which apply the blackhole. For example, given that malicious sources are geolocalized, regional blackholes can be distributed on routers closest to the source of the attacks. An e-commerce website that mostly ships in France, can be partially protected by a European AS that would distribute filtering at edge routers that exchange traffic with other countries. This requires, nevertheless, a geographically distributed network.

Agarwal *et al.* [91] propose to distribute cleaning centers, *i.e.* sets of specialized filtering equipment, inside ISPs. During an attack, the traffic is diverted towards one cleaning center that filters attack traffic. The remaining traffic is redirect towards its initial destination. As cleaning centers can comprise several dedicated filtering equipment, traffic is load-balanced through available equipment inside a cleaning center. Authors provide specifications of such architecture, in terms of hardware and software functions, as well as recommendation on how to implement such architecture. This work paves the way to modern scrubbing centers inside networks. However, multiplying the number of dedicated filtering equipment dramatically increases the overall cost of a cleaning center. Moreover, in some instances, volumetric DDoS attack can consume the cleaning center upstream bandwidth itself, resulting in an inefficient mitigation, as the legitimate traffic is already flooded among malicious traffic before being cleaned.

Combined mechanisms

This section details DDoS mitigation solutions which combines the approaches discussed above. They are mainly based on a list of identified malicious traffic components and bandwidth enforcement mechanisms. Some solutions avoid the overhead of forwarding the malicious traffic to the congested node by using collaboration to filter attacks upstream from the choke point.

D-WARD [93] is an automated mechanism that detect and protect networks against flooding attacks. It is deployed on routers at network as close as possible to the traffic source, *e.g.* at the ISP level. Detection is achieved by monitoring bidirectional traffic, which is possible when implemented at ISP networks for example. Traffic is compared to normal traffic model, so that flows that are not compliant are rate-limited. Although the solution prevents the concentration of attacks flows by filtering close to the source, the target administrator has no means to control the mitigation or at least send feedback.

The Transport-aware IP router [94] is a router that classifies packets based on the transport layer header (transport protocol, TCP flags, ...) at the network edge. It stores the packet class in the IP Differentiated Service (DS) field. It applies different treatments to each class of traffic at edge and core routers, in order to prevent flooding attacks from consuming the whole bandwidth. A bandwidth limit is applied to each aggregate, limiting the bandwidth consumed by the attack. If the attack traffic uses the same transport protocol, *e.g.* a UDP-based amplification attack targets a UDP-based service, legitimate traffic will also be rate-limited.

Intrusion Detection Router [95] also embeds a high-bandwidth flow detection module, which is based on a Counter Bloom Filter (CBF). Packets are added to multiple CBFs and if all counters corresponding to the packet's hash exceed a preconfigured and static threshold, the packet is classified as suspicious. While the mechanism detects high-rate attacks such as SYN floods, it does not distinguish between DDoS and flash crowds (a rush in traffic that only involves legitimate users).

Discussion

The ability of filtering mechanisms to mitigate volumetric DDoS attacks depends on its granularity. A coarse grained mechanism is successfully able to drop malicious traffic. It can however induce large collateral damages by, for example, contributing to the denial of service condition, *i.e.* dropping the legitimate traffic. A fine grained mechanism, which may involve connection tracking or deep packet inspection techniques is mostly able to drop malicious traffic while guarantying a small amount of collateral damages. Such fine grained filtering mechanisms, however, are not embedded in regular equipment and require more resources than coarser mechanisms. As a consequence, they are gathered in filtering bastions, which require diverting mechanisms to convey traffic to mitigation centers.

Scheduling originally intends to decide service (CoT) order and prioritization. It has been transformed into flow fairness enforcement mechanisms. These mechanisms are distributed amongst routers across the net-

work, which allows to act on traffic upstream from the congestion point. Enforcing fairness between flows is not enough in case of a volumetric DDoS attack. Indeed, malicious flows with a fair share further consume bandwidth. As a consequence, a growing number of attack flows induce a reduction of the fair share. The mechanism indiscriminately treats legitimate and malicious traffic.

Queuing mechanisms are intended to prevent congestion by acting proactively, *i.e.* before the congestion collapse. However, in the context of volumetric DDoS attacks, they are reactive, as most mechanisms are intended to enforce a fair use of the queue, *i.e.* equally share the bandwidth between flows in the queue. This means that, during a small period, all flows are allowed to send up to their fair share. Then during a Distributed DoS, bandwidth consumed by the attack traffic is equal to the fair share multiplied by the number of malicious flows.

Policing and shaping guaranty a constrained bandwidth. Mechanisms do not specify themselves to which part of traffic apply the bandwidth constraint. As a consequence, they have to be combined with other mechanisms. The efficiency of system depends on the ability of the preliminary selection of traffic to restrain.

The distribution of coarse grained filtering mechanisms avoids the funneling effect caused by the diversion of traffic towards the mitigation point. It then enhances the resilience of network to volumetric DDoS attack. Moreover, it allows a better management of resources since the configuration of the mechanism can be tuned with regard of the regional context where mechanism has been distributed. However, efficiency of the distributed mechanism is inherent to the granularity of the underlying mechanism.

As combined mechanisms aim at embedding automated detection filtering in network equipment such as routers, they are carefully designed not to consume all equipment resources. Some mechanisms also include cooperation between equipment. The lack of external validation as well as the filtering granularity, induced by resources optimization, may lead to drop legitimate traffic.

2.4 Detour-based mechanisms

Detour mechanisms reflect solutions that are based on the deviation of traffic to primarily alleviate the congestion in a network. Some researchers then extend such mechanisms to volumetric DDoS attack mitigation.

2.4.1 Traffic Diversion

Traffic diversion refers to a mechanism that aims at deflecting traffic towards a middlebox. Once the traffic hits the middlebox, the later can achieve traffic inspection, monitoring or cleaning. Then, if desired, traffic can be rerouted

towards the initial destination. We adopt the same terminology as in [91], *i.e.* diversion refers to the rerouting from the source towards the cleaning center, while redirection refers to traffic from the cleaning center towards the initial destination. Table 2.6 shows different techniques which may be implemented to achieve rerouting.

Rerouting traffic inside an Autonomous System (AS) may be implemented in three different ways.

- At first, diversion is achieved by modifying internal BGP (iBGP) routing announcements, such that the original destination prefix egress point is replaced by the cleaning center address. This announcement is advertised to all routers inside the AS using iBGP mechanisms. In that case, the whole traffic that matches the destination prefix is diverted. Then, traffic is redirected to the original destination through tunnels, which have been beforehand setups. For example, Generic Routing Encapsulation (GRE) protocol [97] or Layer 2 Tunneling Protocol (L2TPv3) [98] can be used to encapsulate traffic in such tunnels. This solution allows using multiple cleaning centers within the AS. The iBGP protocol will then divert traffic from network entry points towards its closest cleaning center. Moreover, it only requires to set up a single tunnel from each cleaning center towards the original destination. Nevertheless, the time before the mitigation is fully operational is constrained by the BGP convergence time, *i.e.*, time to propagate the BGP prefix updates to all routers in the AS. Such solution also requires that tunnel termination is set up, *i.e.*, they are GRE or L2TP enabled.

- Tunnels are set up in all network entry routers in order to divert traffic towards cleaning centers. Routers can use Access Control Lists and Virtual Routing and Forwarding (VRF) table to achieve policy-based routing, a finer traffic selection than destination prefix matching. Traffic is then redirected from the cleaning center towards the original destination using the nominal routing configuration.

- In an MPLS network, an MPLS label is mapped to the path towards the cleaning center, while another one will depict the path from the cleaning center towards the original destination. Both labels may be pre-reserved, such that ingress Label Switching Routers only have to assign the diversion label to traffic to instantiate the rerouting. This solution benefits from MPLS advantages, such as flexibility, ease of traffic engineering, but requires an MPLS infrastructure.

Traffic diversion can be set up between ASes. For example, traffic can be rerouted through an AS, which was not initially in the path from a given source towards the destination. Such rerouting can be implemented in two ways.

	Diversion	Redirection
Inside AS	iBGP announcement	Tunneling (GRE/L2TPv3)
	Tunnels (GRE/L2TPv3)	no iBGP modification
	MPLS tunnel	no MPLS modification
Multi ASes	BGP annoucement	Tunneling (GRE/L2TPv3)
	DNS	No modification

Table 2.6 – Possible combination of techniques used for traffic diversion and redirection

- The AS which is going to receive diverted traffic, announce the destination prefix. As a consequence, it receives all traffic destined to this destination at his network entry points. This AS can then use one of the solutions detailed previously to route traffic inside his network. Traffic is then sent back to the original destination by establishing a tunnel between the cleaning center and the original destination. Like with iBGP, this solution suffers from the time to propagate BGP updates within the Internet. Moreover, ASes do not advertise nor accept prefixes longer than /24, as recommended by the RIPE Network Coordination Centre [99]. As a consequence, a single IP address cannot be diverted using such solution.

- DNS-based diversion is an application-based technique, in the sense that it diverts traffic which destination IP has been preliminarily resolved using a domain name (or subdomain). The mechanism consists of replacing the original A record of the Fully Qualified Domain Name (FQDN) with the public IP of the cleaning center. As a consequence, when clients communicate with this FQDN, they send traffic to cleaning center resolved IP. The cleaning center may then use tunnel techniques to redirect traffic towards the original destination.

  Protecting a site with such a solution requires the site's IP to stay secret. If an attacker knows the real IP of the target, he can directly attack it using the real IP and will not use the FQDN resolution. The time to set up DNS diversion is constrained by the DNS record configuration (Time To Live value), which characterizes the retention of previous DNS resolution.

As the congestion is mainly caused by the funneling effect, diversion-based mitigation may only shift the choke point from somewhere upstream

from the target to upstream from the middlebox. Attack traffic is likely to converge and cause congestion upstream from the mitigation equipment. Moreover, modifying the path from the source to the destination increases the latency of traffic.

2.4.2 Network link usage optimization

Welzl [31] stated that congestion control aims at using the network resources as efficiently as possible, by for example employing all the available capacity. The network link usage optimization starts from the point that during congestion on a link or at a node, other network links can be underused. Given that a volumetric DDoS attack traffic saturates the network at peculiar nodes, *i.e.* the choke point, this subsection reviews the use of network link usage optimization as a countermeasure to volumetric DDoS attack congestion effect.

Multi-path Multipath routing/forwarding is the ability to define multiple path for traffic from a point to a destination. A given traffic is not constrained to a single path, whose bandwidth is overwhelmed, but is distributed over multiple path. Multipath must be able to configure multiple next-hops to this destination. Some researchers [100, 101, 102] use this feature to enhance the network resilience, *i.e.* maintaining sufficient operating level even during attack [103]. The mechanism that shares the traffic between outgoing path, is named *load-balancing*. We address them in the next paragraph.

Multi-path routing is implemented in current routers, for example using OSPF routing protocol [104]. OSPF allows Equal Cost Multi-Path (*ECMP*) routing, *i.e.* alternates path have equal probability to be selected. Menth *et al.* [101] propose the Self-Protecting Multipath (*SPM*) that consists of several disjoint paths that forward traffic before and after a failure of one of the paths. Traffic is re-distributed to the remaining paths in case of outages. Kazmi *et al.* [105] develop a model to select path to provide resilience against link failure, with the aim to reduce link utilization. Murphy and Garcia-Lunes-Aceves [102] use destination-based traffic split together with traffic shaping to avoid congestion and ensure acceptable delays. Using weighted ECMP that load-balances traffic between available paths according to related weights, Zhang *et al.* [106] propose a model to compute split ratios in order to minimize end-to-end delays by optimizing the bandwidth allocation.

IP-based Load-Balancing algorithms In network routing, load-balancing refers to sharing traffic between multiple paths. The share can be achieved in a per-packet fashion such as a using round robin algorithm. Whereas algorithms like weighted round robin scheduling [107] are efficient in terms of computing resources usage, flows may suffer from reordering due to its

random behavior. This leads to packets of the same flow being forwarded through different paths, so that they arrive at the destination in a different order of their departure.

Cao *et al.* [108] study hash-based load-balancing schemes that avoid such a drawback, so that the packets are forwarded to a deterministically path determined by some headers fields such as source or destination IPs. They describe some hash functions used to load-balance the traffic such as XOR of source and destination IPs -based functions or the CRC16 [109] of source and destination IPs and ports.

MPLS Traffic Engineering Traffic Engineering (TE) refers to the mechanisms that allows network administrators to efficiently use network resources. First, it aims at distributing traffic to minimize the network resources usage. Second, it intends to maximize the traffic performance, so that the QoS requirements of traffic are guaranteed. It encompasses a constraint-based path selection, so that, for example, the routing of real-time application traffic can be enforced through paths with small delays.

Multi Protocol Label Switching (*MPLS*) [110] uses MPLS labels to decide where to transmit packets. A label identifies a path across the network (*i.e.* a Label Switching Path - LSP) for a given part of traffic that has to be forwarded in the same manner (Forwarding Equivalence Class - FEC), *i.e.* take the same path with the same per-packet treatment (QoS, ...). Labels are imposed by the MPLS domain ingress routers, named ingress Label Switching Router (*iLSR*) according to packets fields such as the destination IP. MPLS header containing the label is then slipped between the OSI Layer 2 and OSI Layer 3 (or 2 of MPLS network implements a Layer 2 VPN). The header is removed when the packet leaves the MPLS domain. MPLS header could also been stacked, for example to forward an LSP through an MPLS tunnel. In that case, the topmost label is used during the packet switching.

Originally intended to reduce the IP lookup overhead of traditional IP routing, MPLS is mostly used to implement Traffic Engineering and improve network resiliency with failure response mechanisms [111]. Traffic Engineering MPLS extension [112] is intended to ease the configuration of traffic paths and resources assignment. It is based, among other things, on a constraint-based LSP selection for a given FEC. For example, the LSP of a FEC that requires a known amount of bandwidth can be forwarded through routers only if they have sufficient bandwidth in order to prevent oversubscription. The MPLS Differentiated Service [113] extension allows the prioritization of traffic (*i.e.* FEC) during LSP establishment, and packet processing at router (Per-Hop Behavior - PHB).

To preserve legitimate traffic from collateral damage brought upon by attack traffic, Hachem *et al.* [13] proposed HADEGA that uses MPLS support for Differentiated Services and Traffic Engineering. It maps network

flows to several Behavior Aggregates (BA) according to their index of suspicion, *i.e.* their likelihood to be malicious and their impact. Different QoS level are assigned to BAs depending on their level of suspicion. Highly suspicious flows obtain a highly degraded QoS. The QoS assignment to a BA is implemented through MPLS Differentiated Services and per-hop behaviors applied to BAs. The second part of HADEGA uses Traffic Engineering to mitigate the attack. Traffic Engineering allows to establish BA-dedicated LSPs according to administrative topology constraints. As a consequence, volumetric DDoS attack traffic is isolated from the legitimate one and routers can deprioritize (or any other packet processing action such as drop) malicious traffic, so that legitimate traffic is preserved. Authors proposed the use of Snort signatures as a basis for the flow-LSP mapping [114].

2.4.3 Overlay networks

Overlay networks have been introduced by Andersen *et al.* which proposed Resilient Overlay Networks (RON) [115]. They deploy an application layer on top of the existing Internet routing infrastructure. In the normal case, the RON forwarding architecture follows the underlying forwarding plane. By actively monitoring the underlying IP paths, RON nodes detect failures in the underlying data plane and modify its overlay forwarding plane. While authors claimed to resolve most underlying network failure by only adding a single hop to the forwarding plane, their architecture has not been tested against network outages caused by DDoS attacks. The overlay networks have then been actively studied.

Secure Overlay Network (SOS) [116] aim at being a DDoS resilient version of the overlay network. Overlay network edges act as an access point and are responsible for tagging traffic as legitimate or not. This can be achieved using authentication protocols, such as IPsec or TLS. Traffic is then routed through overlay proxies. A proxy takes the decision of transmitting, rate-limiting or dropping the illegitimate traffic. It then redirects traffic towards the destination. The destination filters traffic such that only the packets transmitted by proxies are accepted. SOS requires the deployment of overlay nodes, but also the authentication of all legitimate clients. The cost induced by authentication may dramatically burden the performance of short-lived communication that requires small latency, such as with DNS service. Greenhalgh *et al.* [117] propose an IP-based implementation of SOS, which circumvents the use of a dedicated application protocol for the overlay.

Overlay network are based on the isolation of malicious traffic from the legitimate one. Unlike HADEGA [13], which performs attack traffic identification at the ingress of a network, overlay networks achieve traffic tagging at the target side. Once the volumetric DDoS attack traffic is identified, QoS treatments can be applied to preserve legitimate traffic.

2.4.4 Discussion

Traffic diversion has to be combined with another technique, such as filtering or policing. It allows cooperation between ASes, which may prevent network from being congested.

Network link usage optimization aims at using the whole resources available inside the network. It has limitations, as, considering volumetric DDoS attack, attack traffic may consume the whole network bandwidth. If network has also been overprovisioned (*cf.* Section 2.5.1), network entry points are most likely to be the choke point [118]. In that case, legitimate traffic will suffer from the attack before entering the network.

Overlay networks combine Traffic Engineering, namely to ease reconfiguration of network in case of failure, and Differentiated Service principles, *i.e.* provide different treatment to different types of traffic. This requires a mechanism or an equipment that classifies the traffic in order to determine its overlay. Implementation requires a client-based authentication, which may inefficient when machine is compromised, which is mostly the case when it has been enrolled in a botnet. Although research reduces the requirement for a dedicated application protocol to maintain the overlays, the architecture requirements remains strong, particularly to authenticate traffic.

2.5 Absorbing mechanisms

Absorbing mechanisms aim at handling DDoS attack and legitimate traffic with help of the available resources. This is achieved by either expanding resources (*i.e.* overprovisioning) or modifying resource configuration to make the infrastructure more resilient to flooding attacks that leverage protocol exploitation, especially TCP.

2.5.1 Overprovisioning

Overprovisioning is a straightforward solution against attacks that aim at exhausting resources. It consists of an allocation of resources that is excessive under nominal conditions. Resources are provisioned in such a way that they can handle even malicious traffic [119]. As a result, overprovisioning focuses on mitigating the effect of an attack, instead of the attack itself. The main drawback is that resources are often underutilized, *i.e.* during normal condition. Overprovisioning applies to a wide range of resources, from network to application.

Network Resources

Network resource over-allocation is an old congestion avoidance solution. A well over-proportioned network ensures the continuity of service even during attack, without the overhead of differentiating legitimate from malicious

traffic [120]. It differs from congestion control, which aims at using the network as efficiently as possible. While congestion control mechanisms focus on "how to deal with congestion using the available resources?", overprovisioning allocates more bandwidth until congestion disappears.

Overprovisioning has spread while the decrease of bandwidth price [31]. The cost for overprovisioning bandwidth was then significantly less than the implied price of being passive facing the attack. In 2008, a typical allocation of 3 to 5 times the average bandwidth consumed during legitimate traffic bursts is considered as suitable [121]. However, while the largest bandwidth attack reported in 2008 was 40Gbps, it goes beyond the terabit today. This corresponds to an increase of 2500% of the attack volume. Nonetheless, the Internet transit price only declines by 19% [122], which does not balance the explosion of the attack traffic volume. Moreover, Karagiannis *et al.* [118] argue that congestion does not often occur in overprovisioned parts of networks, *e.g.*, ISP backbones, but on network entry points. As a consequence, while network resources overprovisioning is useful to handle short burst of traffic, *e.g.* during the mitigation set up, maintaining enough resources to mitigate the effect of current volumetric DDoS attacks is too expensive and inefficient.

Application Resources

Overprovisioning application resources is another similar and trivial solution to handle a massive amount of inbound traffic.

Content Delivery Networks Another mean of multiplying service resources is to cache its content and distribute it across the Internet. Content Delivery Networks (CDN) are massively geographically distributed infrastructures which host, entirely or in part, a service content such as HTTP web content. The public IP address of the service is then mapped to the CDN cache IP address. A CDN node forwards requests for content which are not cached to the service.

As a consequence, the end service is only solicited for the update of the cached content or during the request of an uncached content. In return, traffic from the CDN node towards the service is often unconditionally treated as legitimate, such that frequent update or uncached content requests may flood the service. For example, HTTP POST floods consist of requesting dynamic and new content from service (*e.g.*, nonexistent web resources), such that the CDN node is constantly querying the service that will be oversubscribed. Such application overprovisioning should then be combined with a filtering mechanism.

CDN relies on the IP anycast mechanism. It consists of giving the same IP address to multiple endpoints. With the help of routing mechanism, a client will then be directed towards its closest endpoint [119]. The combi-

nation of both enhances the quality of user experience by easing the delivery of content and reducing the latency while decreasing the cost of bandwidth [123].

Anycast reduces the funneling effect of traffic routed towards a single destination, as destinations are disseminated around the Internet. To complete the denial of service condition (*i.e.*, link congestion) attacker has to saturate bandwidth upstream from each CDN point of presence.

Scalable application infrastructure Within the context of emerging cloud-based services, overprovisioning the application itself has become more popular [124]. Virtualization of services allows on-demand allocation and commissioning. Thus, when the number of application requests reaches a static or dynamic threshold, a new virtualized application is instantiated [125]. However, the virtualization infrastructure should be able to handle potential bursts of application demand.

The application maintainer may not own the cloud infrastructure but pay a cloud provider for the use of the computational power [125]. As a consequence, such solution is vulnerable to the economic denial of sustainability attack [126]. This attack increases the cost of maintaining the availability of service. It forces the service provider to pay the cloud platform for more resources required to handle an abnormal amount of requests but does not receive the matching service revenue.

2.5.2 TCP-based mechanisms

TCP attacks leverage specific protocol exploitation. Thus, they require TCP-based mitigation, especially absorbing mechanisms. The TCP stack can be re-configured proactively or in response to an attack.

A key feature of TCP is that it requires connection establishment before upper layers begin to exchange information. The three-way-handshake prevents meaningless application level communication with spoofed sources. The TCP stack requires mechanisms to track connection during and after its establishment. These mechanisms are vulnerable to specific connection depletion attack, such as SYN floods.

SYN floods aim at filling the structure that tracks incomplete connection by overwhelming the server with SYN packets, *i.e.* connection attempts. As a consequence, new legitimate connection attempts are discarded or replaced with malicious ones. TCP stack parameters can first be tuned to reduce the vulnerability to such attacks.

From a Linux-based server perspective, the connection establishment function as follows (other implementations are similar). The server receives a SYN packet from a client. It then instantiates a Transmission Control Block (TCB) for this new connection into the backlog (queue reserved to

incomplete connections) and sends a SYN-ACK to the client to acknowledge the reception of the connection initialization. The server then sets the TCB state to SYN-RECEIVED and waits for the client acknowledgment to complete the connection initialization. Once it receives the ACK from the client, the TCP connection is established, such that its TCB state is set to ESTABLISHED and moved from the backlog to the queue of established connections. If the server does not receive ACK within a predefined period (timeout), it resends a SYN-ACK. After few retransmissions of the SYN acknowledgment, the TCB is recycled for other incoming connection attempts.

As the backlog size, *i.e.*, the number of TCB, is fixed, an attacker can send a sufficient amount of SYN packets to fill the backlog with incomplete SYN-RECEIVED TCB. A legitimate client would then have no means to initiate a connection. A first straightforward countermeasure would be to increase the backlog size [127]. This, nevertheless, increases the rate the server can accept TCP connection. The application should have enough resources to handle as many connections if all sources complete the handshake. Moreover, backlog size increasing is constrained by the server resources and cannot be infinite. As a consequence, an attacker may still send SYN packets at a rate that is sufficient to fulfill the backlog.

Another simple solution is to reduce the time before a TCB is flushed for not receiving the ACL from the client (SYN-received timer) [127]. So that, incomplete connections are reaped more quickly and the SYN packet acceptance rate increases. Clearly, the timer should not be less than the Round Trip Time (RTT), *i.e.*, time for the SYN-ACK to reach the client and the ACK response to be received by the server. However, even if this guideline is applied, massive attacks that cause congestion can force clients to retransmit the SYN-ACK. The retransmission may then be discarded by the server due to the reduction of the SYN-received timer. Bekravi *et al.* [128] combine both parameters reconfiguration and propose an automated method to choose parameter values.

2.5.3 Discussion

Overprovisioning mechanisms look similar to the cat and mouse game. It is based on the assumption that attacker has insufficient resources to overload the victim resources. Nonetheless, in the context of volumetric DDoS attacks, this only shifts the choke point to another resource or equipment as, for example, congestion occurs on other parts of networks [118]. These mechanisms should then be combined with another mitigation such as filtering mechanisms.

TCP-based mechanisms are not applicable to UDP-based volumetric DDoS attack. Given that those mechanisms aim at mitigating incomplete TCP-handshake floods (*e.g.* SYN floods), they are also inefficient for stateful

TCP-based applicative floods such as HTTP GET floods.

2.6 Conclusion

This chapter introduced the area of Distributed Denial of Service attacks. We then presented the volumetric Distributed Denial of Services attacks and detailed some of their characteristics, namely victims, method of attack and sources. Different metrics to measure the service disruption have been summarized. We then differentiated network-based metrics from those that take into account the human perception. We finally classified countermeasures to DDoS into three main categories, namely admission control, detour-based and absorbing mechanisms. The first decide whether a communication (*e.g.* packet, request) should be forwarded through its destination or not. The decision can be may require client to perform a kind of authentication (active admission control) or not (passive admission control). The second make use of network path modification to redirect traffic to a cleaning device (traffic diversion technique), make better use of network resources (network link usage) or isolate malicious traffic from legitimate one (overlay network). The third refers to the overprovisioning of resources or specific TCP-based techniques to mitigate TCP-based attacks.

Multipath routing is a promising volumetric DDoS attacks damage control method. It allows conveying both legitimate and attack traffic through the network. In fact, it avoids the funneling effect of such attacks by forwarding traffic towards targets through several path. If the choke point cannot be prevented using multipath routing, it can be combined with differentiated forwarding and Quality of Service treatments. Legitimate traffic is then going to be prioritized along the network and won't longer suffer from the high network resource consumption by the attack traffic. However, it requires suspicious traffic to be isolated from legitimate traffic so that different treatments can be applied to each. Conversely to Overlay Networks that require traffic to be authorized by the end service to perform traffic segregation, the isolation of malicious can be given at the network entry using passive admission control mechanisms

As blacklists are widely used in real environments [129, 130, 131], in Chapter 4, we study the efficiency of this peculiar passive admission control mechanism in operational environment. We particularly, assess the impact of information availability on the ability to filter attack traffic and to limit collateral damages.

Chapter 3

Load-Balancing based Mitigation Technique

Contents

Multipath routing and especially load-balancing mechanisms are commonly implemented functions in off-the-shelf network equipment. They aim at mitigating network congestion by making better use of network resources, by benefiting from efficient routing functions, *i.e.* without the overhead of traffic filtering. In return, the mechanism has no means to differentiate legitimate from malicious traffic on its own. Multipath routing has then to be combined with a filtering tool, such that a differentiated treatment can be applied to both traffic types. As volumetric DDoS attacks deal with huge amount of traffic, we expect achieving malicious traffic isolation with a minimum overhead to routing equipment.

We introduce the principles and architecture of a mitigation based on multipath routing. We then evaluate the usability of commonly used multipath techniques to achieve the filtering function, *i.e.* to coarsely segregate attack and legitimate traffic and forward them through different paths. We focus on an off-the-shelf Cisco router and on literature advocating MPLS load-balancing mechanisms. The objective is to limit the impact of a voluminous attack could have on the legitimate traffic through saturation of network resources. In this thesis, as we have been unable to achieve this goal, we designed a deterministic load-balancing scheme.

We then proposed a new MultiProtocol Label Switching (MPLS) load-balancing mechanism to achieve such damage control for volumetric DDoS attacks. Our mechanism acts as a coarse grained mitigation which only prevent network resources, especially links, from being overwhelmed. We then present a DDoS mitigation mechanism dispatching suspicious and legitimate traffic into separate paths, well upstream from the target. The separation of traffic is based on a signature identifying suspicious flows, carried in an MPLS label, and then used by a load-balancing mechanism in a router. We benefit from the MPLS extensions, such as Traffic Engineering and Differentiated Services, which provide resource aware multipath routing and differentiated forwarding treatments. The legitimate traffic is then preserved at the expense of suspicious flows, whose resource allocations are throttled as needed to avoid congestion. Suspicious traffic can be fine-filtered inside the network using, for example, a dedicated solution. This prevent distributing expensive dedicated middle-boxes as suspicious traffic is throttled upstream from the middle-box. The combination of both coarse and fine grained mechanism is not addressed in this thesis.

3.1 Mitigation Approach

In this section, we present the principles of a load-balancing based damage control mechanism for volumetric DDoS attack. In other words, it acts as coarse mitigation which only prevents network resources, especially links, from being overwhelmed. We prioritize the forwarding of legitimate traffic, such that malicious traffic suffers first from bandwidth limitation. We do not intend to clean the traffic from its malicious component.

Figure 3.1 shows our generic damage control network architecture. At each network access point, traffic towards a given target is load-balanced across at least two paths, namely prioritized (dotted line) and unprioritized (dashed line) links. Prioritized link finally reaches the target. But before, unprioritized or both unprioritized and prioritized links may be diverted towards scrubbing centers to achieve a complete traffic cleaning. As traffic sent by the middle-box is considered as cleaned, it is forwarded through a prioritized link.

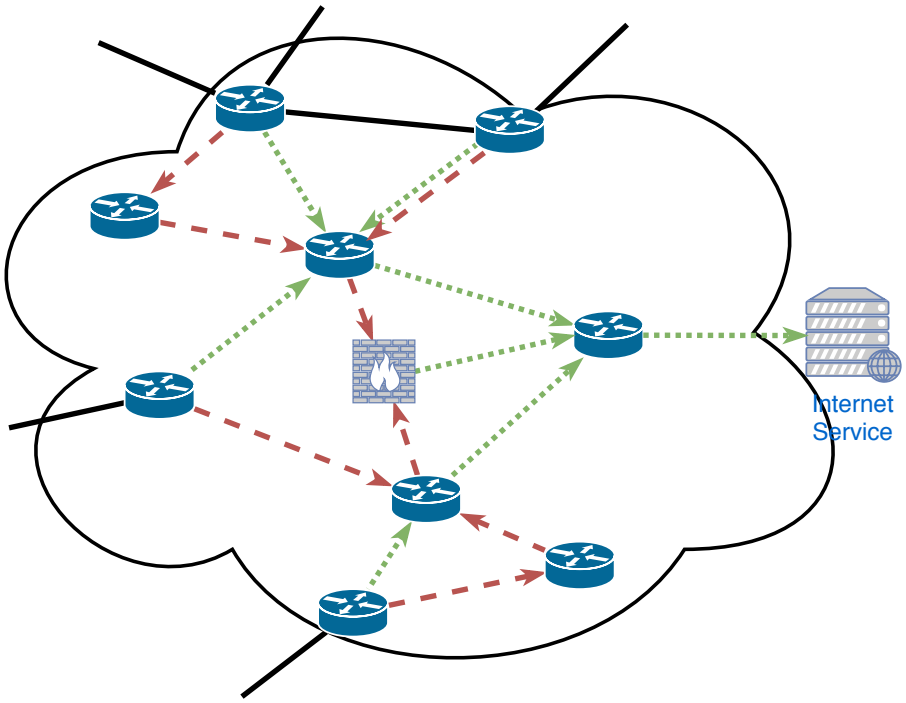


Figure 3.1 – Multipath volumetric DDoS attack mitigation principle

Both prioritized and unprioritized paths do not necessarily refer to physical (or IP layer) links. For instance, both can follow and share the same physical link from a given access point towards the endpoint. An IP layer segment can also contain other traffic, *e.g.*, towards several destinations. As

a consequence, routers have to identify whether a packet belongs to either prioritized or unprioritized. To this end, the entry router has to identify traffic, whether it is legitimate or not, and forward the identifier together with the traffic. For example, it can forward each type of traffic through a dedicated tunnel (one for each type of traffic) or map them to a different MPLS label or apply a different mark onto the IP Differentiated Services Code Point (DSCP) header field. The IP DSCP header is a specific field used to apply differentiated treatments to services. Core routers then apply differentiated quality of service treatments for each type of traffic, according to the identification made and forwarded by the entry router. Both entry and core routers also should constrain at least the unprioritized traffic to a given bandwidth. Thus, routers can maintain the bandwidth allocated to other traffic types.

In this chapter, we focus on the load-balancing method that redirects as much malicious traffic as possible through an unprioritized path while letting the rest of traffic flow through another one. We then consider a single network entry point, from which traffic will be split among two paths. Nonetheless, we do not address the routing of traffic inside the network toward the destination, *i.e.*, which core routers will be successively traversed by each traffic towards the end point. We focus on the traffic segregation at the network entry points.

Following that we deal with IP-based traffic and we do not make an assumption on the inner (Layer 4) protocol, we reduce load-balancing to the IP layer. In the remainder of this section, we study existing IP-based load-balancing mechanisms to achieve our goal.

3.2 Off-the-shelf Cisco Router Load-Balancing

In this section, our purpose is to use off-the-shelf routers to achieve a deterministic load-balancing. In fact, each packet of a given communication (*e.g.*, determined by the source and destination IP addresses) need to be forwarded through the same path, *i.e.* sequence of routers, such that load-balancing mechanisms have to deterministically select the outbound path accordingly. A load-balancing router also has to deterministically forward in a similar way a major part of malicious communication, so that specific forwarding treatments (*e.g.* unprioritized QoS treatments) can be applied to. Thus, we investigate, whether this can be achieved using off-the-shelf equipment. Tests are performed with a real Cisco 7200 router. It benefits from the version 12.4 of the Internetwork Operating System (IOS) firmware.

3.2.1 Cisco 7200 load-balancing principles

Our router implements Cisco Express Forwarding (*CEF*) mechanism, which accelerates the routing and forwarding processing by caching (*e.g.*, in hard-

ware) routing information. The distribution of packets amongst multiple links is based on the CEF mechanisms. Our router has two deterministic load-balancing algorithms, namely original and universal [132]. The original algorithm uses a hash of the source and destination IP addresses to balance traffic between outbound paths. As a consequence, each router load-balances traffic in the same way, which may induce an unwanted behavior in the network known as polarization [133]. For example, Figure 3.2 shows successive routers that distribute traffic using the original algorithm. A communication that is load-balanced through the second link may always be distributed through the second link on each router if the same context applies, *e.g.* link weights.

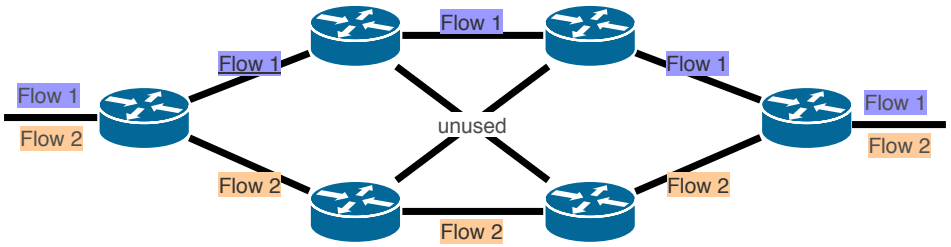


Figure 3.2 – CEF polarization effect [133]

The universal algorithm aims at circumventing this drawback [132]. By introducing a salt in the hash key, it allows routers to distribute a communication differently. The salt is configurable via an ID, *cf.*, Listing 3.1. For a given set of malicious connections (defined by the source and destination IP addresses), we want to find an ID such that a major part of those malicious connections are forwarded through a given outbound path. We expect that the number of malicious connections forwarded through the dedicated link is greater than the load-share proportion, *i.e.* $\frac{1}{\text{number of links}}$. Both algorithms share the load in the same way. Routers globally combine routing information (*e.g.*, number of links, link weights) to compute a load distribution.

Listing 3.1 – Configuration of the universal load-balancing algorithm

```
r2-7204vxr(config)#ip cef load-sharing algorithm universal ?
  <1-FFFFFFFF>  Fixed ID
  <cr>
```

Listing 3.2 shows the distribution of communications destined to the prefix 10.0.0.0/8. Two outbound gateways are configured for this prefix, namely 192.68.2.2 and 192.168.1.2 (*cf.* lines 7 and 11). The load distribution maps a packet hash result to an outbound link as shown from lines 20 to 35. Then, for each packet, routers produce a 16 bit hash from the input key

generated using packet fields. The hash is then mapped to a next-hop with the help of the load-distribution.

Listing 3.2 – Cisco router Load-distribution

```
r2-7204vxr#show ip cef 10.0.0.0 internal
10.0.0.0/8, version 81, epoch 0, per-destination sharing
0 packets, 0 bytes
  Flow: AS 0, mask 8
  via 192.168.2.2, 0 dependencies, recursive
    traffic share 1
    next hop 192.168.2.2, GigabitEthernet0/2 via 192.168.2.2/32
    valid adjacency
  via 192.168.1.2, 0 dependencies, recursive
    traffic share 1
    next hop 192.168.1.2, GigabitEthernet0/1 via 192.168.1.2/32
    valid adjacency

  0 packets, 0 bytes switched through the prefix
  tmstats: external 0 packets, 0 bytes
           internal 0 packets, 0 bytes
  Load distribution: 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 (refcount 1)

  Hash  OK  Interface                 Address         Packets
  1     Y   GigabitEthernet0/2        192.168.2.2           0
  2     Y   GigabitEthernet0/1        192.168.1.2           0
  3     Y   GigabitEthernet0/2        192.168.2.2           0
  4     Y   GigabitEthernet0/1        192.168.1.2           0
  5     Y   GigabitEthernet0/2        192.168.2.2           0
  6     Y   GigabitEthernet0/1        192.168.1.2           0
  7     Y   GigabitEthernet0/2        192.168.2.2           0
  8     Y   GigabitEthernet0/1        192.168.1.2           0
  9     Y   GigabitEthernet0/2        192.168.2.2           0
  10    Y   GigabitEthernet0/1        192.168.1.2           0
  11    Y   GigabitEthernet0/2        192.168.2.2           0
  12    Y   GigabitEthernet0/1        192.168.1.2           0
  13    Y   GigabitEthernet0/2        192.168.2.2           0
  14    Y   GigabitEthernet0/1        192.168.1.2           0
  15    Y   GigabitEthernet0/2        192.168.2.2           0
  16    Y   GigabitEthernet0/1        192.168.1.2           0
  refcount 4
r2-7204vxr#
```

3.2.2 Study Approach

We aim at modeling the universal load-balancing mechanism, in order to determine whether it can be used to maximize the number of malicious connections forwarded through a dedicated outbound next-hop. We thus try to reverse the universal algorithm. However, reversing such black box with this number of inputs (salt, source, and destination IP addresses) seems too difficult. We then assumed that the universal algorithm is an extension

of the original algorithm, such that we can begin by reversing the original algorithm.

We first configure a simple Equal Cost Multipath on the router. Listing 3.3 show the configured routing table. Two gateways are set up for the 10.0.0.0/8 destination prefix, *i.e.*, 192.168.1.2 and 172.16.2.2 (*cf.* lines 14 and 15). The two next-hops are forwarded through respectively interfaces GigabitEthernet 0/1 and 0/2 as shown in lines 16 and 17.

Listing 3.3 – Experimental Routing Table

```
1  r2-7204vxr#show ip route
2  Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
3         D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
4         N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
5         E1 - OSPF external type 1, E2 - OSPF external type 2
6         i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS level-2
7         ia - IS-IS inter area, * - candidate default, U - per-user static route
8         o - ODR, P - periodic downloaded static route
9
10 Gateway of last resort is 192.168.2.2 to network 0.0.0.0
11
12       2.0.0.0/32 is subnetted, 1 subnets
13 C        2.2.2.2 is directly connected, Loopback0
14 S     10.0.0.0/8 [1/0] via 192.168.2.2
15                  [1/0] via 192.168.1.2
16 C     192.168.1.0/24 is directly connected, GigabitEthernet0/1
17 C     192.168.2.0/24 is directly connected, GigabitEthernet0/2
18 S*    0.0.0.0/0 [1/0] via 192.168.2.2
19 r2-7204vxr#
```

We then use the router command line interface to query the exact outbound path, which will be followed by a given source, destination IP addresses pair. The *show ip cef exact-route* (*cf.*, Listing 3.4) command returns the outbound path or interface and the hash result, in the form of the bucket value ranged from 0 to 15. Retrieving a sufficient number of input-output pairs allows us to guess the original load-balancing function.

Listing 3.4 – Cisco command line interface query for the exact-route followed by a given source destination IP addresses pair

```
r2-7204vxr>show ip cef exact-route 0.0.0.0 10.0.0.1 internal
0.0.0.0          -> 10.0.0.1        : GigabitEthernet0/2 (next hop 192.168.2.2)
                                      Bucket 2 from 16, total 2 paths
r2-7204vxr>
```

3.2.3 Results

Table 3.1 shows the mapping between the input of the load-balancing function (source and destination IP addresses) and the hash value. As the load-sharing distribution (*cf.* Listing 3.2) ranges values from 1 to 16 instead of 0 to 15 as provided by the exact route query (*cf.* Listing 3.4), tables 3.1a to 3.1d provide a mapping of the hash values between 1 and 16. More specifically, Table 3.1a shows the mapping of flows connections towards the 10.0.0.1 IP address to the hash. We note that the third topmost byte of source IP address does not impact the hash value. For example, the connection from 0.0.0.1 to 10.0.0.1 gets the same hash result as the connection from 0.0.1.1 towards

10.0.0.1. Table 3.1b, which depicts the mapping of connections towards the 10.1.0.1 IP address to the hash, shows similar trends. Hashes with a value equal to "-" refers to untested source and destination IP addresses pairs. Nonetheless, we show them in order to visually highlight the relationship between IP pairs and hash values.

We succeed in manually reversing the original algorithm. Equation (3.3) depicts the hash function, where $s_0.s_1.s_2.s_3$ and $d_0.d_1.d_2.d_3$ respectively refers to the source and destination IP addresses. The rotleft function also refer to the function which rotates bits to the left by a given number of positions. We note in particular that the two most significant bytes of both source and destination addresses had no impact on the hash value. In fact, when load-sharing a subnet, the most significant bytes that refer to the network part do not introduce entropy in the hash computation.

$$hash(s_0.s_1.s_2.s_3, d_0.d_1.d_2.d_3) = Hsrc(s_0.s_1.s_2.s_3) \wedge Hdst(d_0.d_1.d_2.d_3) \tag{3.1}$$

$$Hsrc(s_0.s_1.s_2.s_3) = ((s_3 \wedge s_4) \gg 1) \ \& \ \text{0xF} \tag{3.2}$$

$$Hdst(d_0.d_1.d_2.d_3) = (((d_3 \ll 1) \wedge \text{rotleft}(d_4, 1)) \ \& \ \text{0xF}) \wedge (s_3 \gg 4) \tag{3.3}$$

Nonetheless, we did not succeed in reversing the universal algorithm as an extension of the original algorithm. Moreover, the potential gain of the use of the universal salt is limited. In fact, the load-sharing algorithm is configured for the entire router. As a consequence, protecting multiple targets (*i.e.*, destination prefixes) requires to find a salt which fits into multiple contexts:

- a router can be traversed by several attack traffics, each with its own set of malicious connections and targets;
- each target can be located in a different location inside or outside the network, so that outbound paths of routers have to be customized for each attack traffic, as well as path weights, ...

We then abandoned the idea of using such load-balancing algorithm.

3.3 MPLS-based Load-Balancing

Our goal is to achieve a legitimacy-based redirection of traffic towards an unprioritized path for attack traffic and a prioritized path for the remaining traffic. The HADEGA [13] solution sounds promising to provide a backend for paths establishment and prioritization. In this section, we first introduce MPLS concepts. We then investigate MPLS-based load-balancing mechanisms.

source IP	hash	source IP	hash
0.0.0.0	3	0.0.1.0	3
0.0.0.1	3	0.0.1.1	3
0.0.0.2	4	0.0.1.2	4
0.0.0.3	4	0.0.1.3	4
0.0.0.4	1	0.0.1.4	1
0.0.0.5	1	0.0.1.5	1
0.0.0.6	2	0.0.1.6	2
0.0.0.7	2	0.0.1.7	2
0.0.0.8	7	0.0.1.8	7
0.0.0.9	7	0.0.1.9	7
0.0.0.10	8	0.0.1.10	8
0.0.0.11	8	0.0.1.11	8
0.0.0.12	5	0.0.1.12	5
0.0.0.13	5	0.0.1.13	5
0.0.0.14	6	0.0.1.14	6
0.0.0.15	6	0.0.1.15	6
0.0.0.16	11	0.0.1.16	11
0.0.0.17	11	0.0.1.17	11
0.0.0.18	12	0.0.1.18	12
0.0.0.19	12	0.0.1.19	12
0.0.0.20	9	0.0.1.20	9
0.0.0.21	9	0.0.1.21	9
0.0.0.22	10	0.0.1.22	10
0.0.0.23	10	0.0.1.23	10
0.0.0.24	15	0.0.1.24	15

(a) Destination IP = 10.0.0.1

source IP	hash	source IP	hash
0.0.0.0	1	0.0.1.0	1
0.0.0.1	1	0.0.1.1	1
0.0.0.2	2	0.0.1.2	2
0.0.0.3	2	0.0.1.3	-
0.0.0.4	3	0.0.1.4	-
0.0.0.5	3	0.0.1.5	-
0.0.0.6	4	0.0.1.6	4
0.0.0.7	-	0.0.1.7	-
0.0.0.8	5	0.0.1.8	-
0.0.0.9	-	0.0.1.9	-
0.0.0.10	6	0.0.1.10	-
0.0.0.11	-	0.0.1.11	-
0.0.0.12	7	0.0.1.12	-
0.0.0.13	-	0.0.1.13	-
0.0.0.14	-	0.0.1.14	-
0.0.0.15	-	0.0.1.15	-
0.0.0.16	9	0.0.1.16	-
0.0.0.17	-	0.0.1.17	-
0.0.0.18	10	0.0.1.18	10
0.0.0.19	-	0.0.1.19	-
0.0.0.20	11	0.0.1.20	-
0.0.0.21	11	0.0.1.21	-
0.0.0.22	12	0.0.1.22	-
0.0.0.23	-	0.0.1.23	-
0.0.0.24	-	0.0.1.24	-

(b) Destination IP = 10.1.0.1

dest. IP	hash	dest. IP	hash
10.0.0.0	1	10.0.4.0	9
10.0.0.1	3	10.0.4.1	11
10.0.0.2	5	10.0.4.2	13
10.0.0.3	7	10.0.4.3	15
10.0.0.4	9	10.0.4.4	1
10.0.0.5	11	10.0.4.5	
10.0.0.6	13	10.0.4.6	
10.0.0.7	15	10.0.4.7	
10.0.0.8	1	10.0.4.8	9
..			
10.0.0.15	15	10.0.4.15	7
0.0.0.18	10	0.0.1.18	10
0.0.0.20	11	0.0.1.20	-
0.0.0.21	11	0.0.1.21	-
0.0.0.22	12	0.0.1.22	-

(c) Source IP = 0.0.0.0

source IP	dest. IP	hash
10.255.255.255	10.0.0.0	1
10.0.0.0	10.255.255.255	15
10.26.52.48	10.37.142.72	7
10.37.142.72	10.26.52.48	9
0.0.52.48	10.0.14.72	15
0.0.14.72	0.0.52.48	9
0.0.52.48	10.0.142.72	7
0.0.52.48	10.0.114.72	2

(d) Random source-destination IP pairs

Table 3.1 – Cisco original load-balancing algorithm hash values (not exhaustive)

3.3.1 MultiProtocol Label Switching

The proposed mitigation solution builds on top of the MultiProtocol Label Switching (*MPLS*) [110]. MultiProtocol Label Switching is a forwarding paradigm based on label switching operations. The goal of MPLS is to offload the IP lookup to the network ingress router (ingress Label Switching Router - *iLSR*) which then label packets. Hence, core routers (Label Switching Routers - *(*LSRs)) forward packets only by looking at the label. As a consequence, LSRs do not take the decision of where to forward packets. Instead, they switch packets according to a path which have been beforehand established for a given Forwarding Equivalence Class (*FEC*). A FEC defines a part of traffic which has to be forwarded in the same manner, *i.e.* it follows the same path (source-destination pair) which the same forwarding treatment. The MPLS headers, including the label, are pushed between the layers 2 and 3 of the OSI model.

We use MPLS to leverage its two existing extensions: 1) Traffic Engineering [112] that allows managing and optimizing network resources. We use this extension to establish mitigation and priority LSPs. Both LSPs can be established either before the attack detection or during the mitigation set up. 2) Differentiated Services [113] that allows classifying traffic and defining bandwidth constraints for each path at the reservation time. In fact, it enables us to forward the flows while prioritizing legitimate traffic and to prevent attack traffic from impacting other traffic [13]. The bandwidth constraints of LSPs should be carefully selected so that the total volume of the traffic going to the priority and mitigation links does not exceed the capacity of the egress router or the on-premise mechanism. One way to achieve this is to setup bandwidth throttling using MPLS extensions for both priority and mitigation LSPs.

From a business point of view, a network operator can approach the bandwidth allocation in different ways. He can reserve a certain amount of bandwidth for mitigation paths of all customers and dynamically assign a portion for a customer under attack. In another situation, the mitigation path could be permanently allocated to a customer who may have subscribed to the service. In both use cases, the network service provider carefully chooses the mitigation allocation as a tradeoff between the cost of bandwidth and mitigation efficiency. We address this question in Section 3.6.

3.3.2 Labels dedicated to Load-Balancing

By default, routers achieve MPLS traffic load-balancing using IP-layer fields [134]. However, the MPLS header does not embed information about his inner payload. As a consequence, in order to parse inner IP-layer headers, routers need to guess the protocol encapsulated in MPLS and check that it matches either IPv4 or IPv6 protocol. They achieve that by parsing the first byte after the

MPLS header and compare to the first byte of an IPv4 or IPv6 header, which contains the IP protocol version. However, routers can mistakenly interpret this first byte in case of a non-IP payload [135]. Such misinterpretation may lead to inconsistent load-sharing across outbound links and degrade the delivery of traffic, *e.g.*, caused by packets re-ordering. Flow-Aware Transport (FAT) [136] and Entropy Label (EL) [137] are standardized MPLS labels which address this load-balancing issue in MPLS network.

The use of FAT labels has been defined for Layer 2 (*e.g.*, ethernet) traffic encapsulated within an MPLS header (*i.e.*, pseudo wire). As shown in Figure 3.3, a FAT label is pushed onto the bottom of the MPLS label stack, between the pseudo wire label and the MPLS payload. The MPLS edge router that encapsulates the Layer 2 frame also computes and inserts the FAT label. It generates the FAT label by using Layer 2 and Layer 3 fields, in such a way that flows that follow the same path get the same FAT label. Load-balancing is then performed using MPLS labels on the stack or the last FAT label, depending on the implementation.

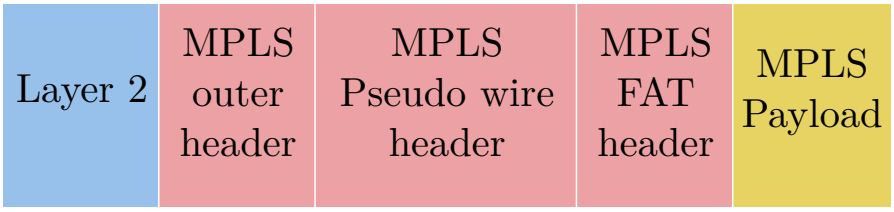


Figure 3.3 – MPLS datagram containing a Flow Aware Transport (FAT) label

FAT labels have been extended and generalized to all MPLS encapsulated payload (Layer 2, Layer 4 or MPLS payload) by Entropy Labels. Figure 3.4 shows a datagram of a frame containing an EL. Before pushing the transport label, a Label Switching Router (LSR) can insert the Entropy Label Indicator - a Special Purpose Label [138] that signals the presence of an EL on the next label to downstream LSRs - and the EL itself. Transit LSR that performs load-balancing will then make use of the EL in the same way as for FAT labels.

Transit routers no more require to guess the IP headers to perform load-balancing of IP-based traffic. It then suppresses the risk of IP header misinterpretation and then QoS degradation. Nevertheless, FAT and Entropy labels pave the way to labels specially crafted to perform load-balancing. We then intend to compute an MPLS label that will serve the load-balancing goal.

Figure 3.4 – MPLS datagram containing an Entropy Label (EL)

3.3.3 Load-Balancing Mechanism

Cyclic Redundancy Check is a commonly used error detection mechanism. CRC 16 bits is an appropriate algorithm to consider in our study, because it is also cited in several papers [108, 139] and patents [140, 141, 142] as a potential load-balancing technique.

In such a hash-based load-balancing algorithm, a subset of packet header fields (*e.g.* source address, destination address, source port, destination port and Layer protocol ID) are concatenated and provided as a key to the hash function [143]. How the hash result is handled to select the outbound path depends on the vendor implementation.

Once the packet hash is computed, it is mapped to the outbound link. As advertised in Figure 3.5 (extracted from [108]), weighted load-sharing amongst N links is achieved using a simple hash table. The size of the table is equal to M, the sum of all potential outbound link weights w_i. As we consider only 2 outbound links in our case study, the hash table can be summarized by the mapping function shown in Equation (3.4).

$$out(hash) = \begin{cases} link1 & \text{if } hash\%M < w_1 \\ link2 & \text{otherwise} \end{cases} \tag{3.4}$$

3.3.4 Experimental Approach

In this section, we study how the CRC16 function behaves when hashing the load-balancing label and the source IP address of a packet. Our use case considers a simple attack which targets a single destination IP address. Thus, for now, we do not use the destination IP address in the hash key.

Our purpose is to find a label that is able to maximize the number of malicious IPs load-balanced through the unprioritized path. Our experimental approach is summarized as follow. We first generate 2000 malicious random source IP addresses. This number is arbitrarily introduced to quickly validate or reject our approach. We then brute force all potential 20 bits labels, *i.e.*, which value is between 16 and 1048575 ($2^{20} - 1$). Label values 0 to 15 are reserved. For each potential label, we emulate the load-balancing of a

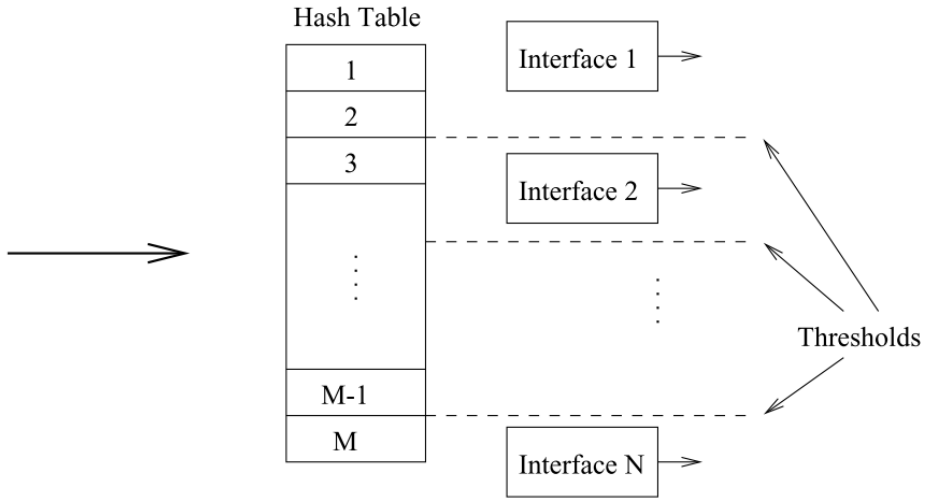


Figure 3.5 – Table-based Hashing with Thresholds Mapping [108]

packet using the label and the source IP among those randomly generated. We count IP addresses forwarded through the unprioritized path and select the label which maximizes this count. We replay this experiment 20 times with different sets of random malicious source IPs. We also repeated each set of experiment with different weights distribution between prioritized and unprioritized paths.

3.3.5 Experiments

Figure 3.6 shows the mean, minimum and maximum number of true positives, *i.e.*, malicious source IP addresses which have been forwarded through the prioritized path. The different values are computed over the 20 experiments replays. The x-axis depicts the different load-shares. For example, a *1 : 3* load-share represents respectively weights of 1 and 2 (*i.e.* $3 - 1$) assigned to the unprioritized and prioritized paths. Blue bars refer to the results of the experiments, while orange bar shows probabilistic results. Weighted hash-based load-balancing are probabilistic mechanisms that forward source IP (in our case) through a path proportional to the weight assigned to this path. So that the probabilistic value is computed by Equation (3.5), where $\#malicious\_IPs$ is equal to 2000. $unprioritized\_weight$ and $sum\_weights$ respectively denote the unprioritized link weight and the sum of all link weights.

$$\#malicious\_IPs \times \frac{unprioritized\_weight}{sum\_weights} \tag{3.5}$$

Globally, considering a given load-share, the average number of true

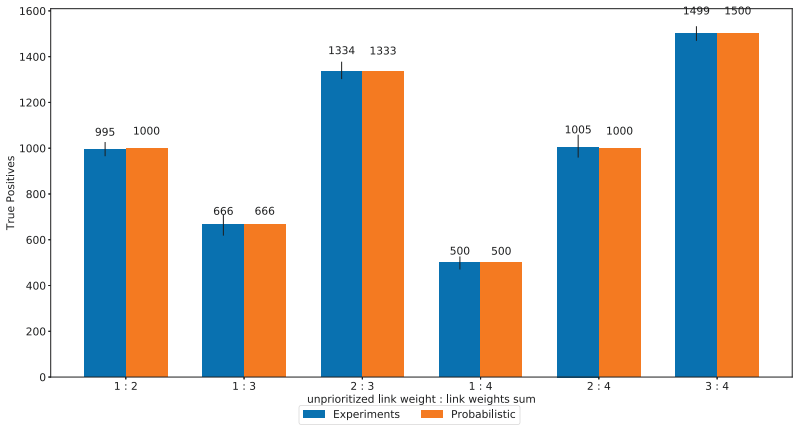


Figure 3.6 – True positive count of a CRC16-based load-balancing based on an MPLS label and the source IP address (2000 malicious IPs)

positives are close to the probabilistic value. Also noteworthy is that for all load-share the minimum count of true positive is below the probabilistic value. This highlight that, at least for one experiment replay, none of the potential labels have been able to forward more than the probabilistic count of malicious source IPs through the unprioritized path. Thus, in those case, load-balancing has been unable to load-balance more than the load-share proportion as expected in the normal case. In other words, it does not provide added value compared to an off-the-shelf load-balancing algorithm.

In the next section, we then provide details on a proposed load-balancing mechanism that fulfill our goal.

3.4 Mitigation Label: a MPLS -based DDoS mitigation mechanism

We replace CRC16èbased hash function by a novel load-balancing mechanism, that distributes flows among quality of service-aware MPLS paths in order to mitigate volumetric DDoS attacks. Legitimate and attack traffics are forwarded through a high and low priority path respectively. The separation mechanism is built upon a lightweight signature identifying traffic responsible for congestion downstream.

Section 3.4.1 lays the foundation of our load-balancing based damage control solution. We then introduce a new signature scheme generation process required for load-balancing traffic in Subsection 3.4.2. In Subsection 3.4.3, we expose the composition of the Mitigation Label and exposes the load-balancing process.

3.4.1 Concepts and Architecture

The overall architecture is depicted in Figures 3.7 to 3.10. In Figure 3.7, a monitoring system continuously collects metrics from network equipment.

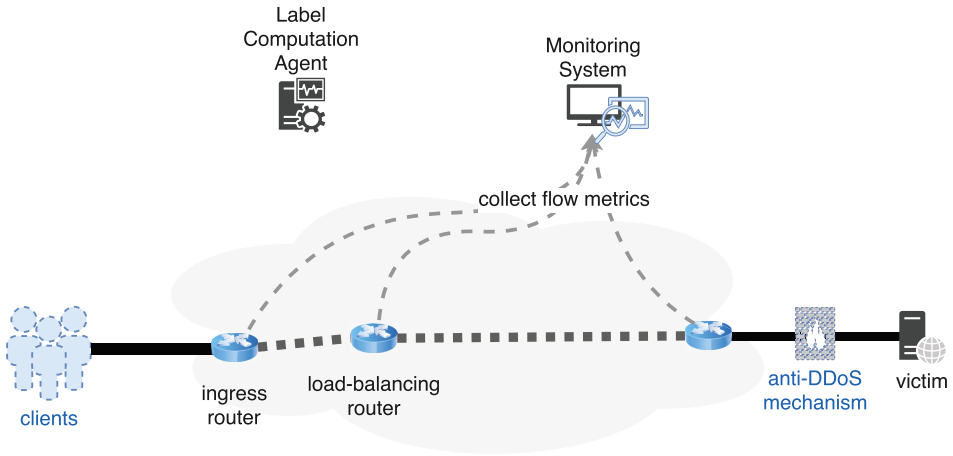


Figure 3.7 – Nominal state without attack

It is also aware of DDoS events in the network, either by detecting the attack itself or by collecting alerts from dedicated sensors, as shown in Figure 3.8.

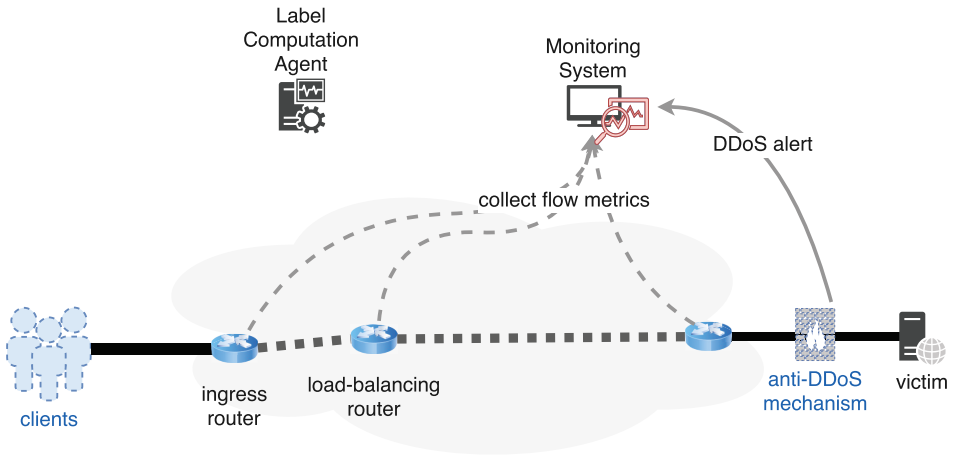


Figure 3.8 – Detection of attack

The mitigation is triggered by the monitoring system sending or transferring a DDoS alert to a Label Computation Agent (*LCA*) (*cf.* Figure 3.9). The agent processes the alert and, in order to finalize the mitigation configuration, may request additional flow metrics (Section 3.4.1) from the monitoring system. Then, the LCA generates a *Mitigation Label* that contains

load-balancing parameters (Subsection 3.4.2) for each network ingress LSR (*iLSR*) involved in the mitigation (only one is shown in figure). The LCA then advertises the Mitigation Label to each iLSR, that in turns push the Mitigation Label onto the MPLS label stack of packets belonging to the sFEC. The sFEC is a MPLS Forwarding Equivalence Class (FEC) containing flows involved in volumetric attack. In other words, sFEC is intended to group flows responsible for congestion. The Mitigation Label is then forwarded together with the packet (see up to the load-balancing router.

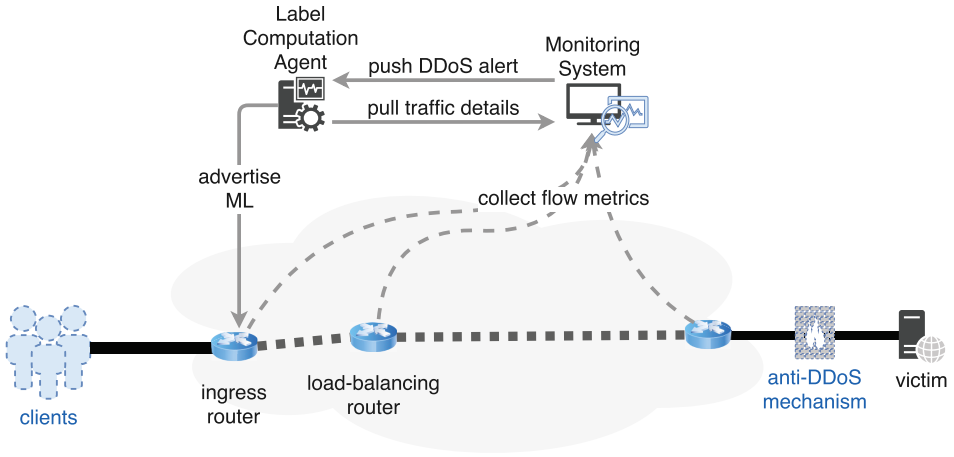


Figure 3.9 – Computation of attack signature and mitigation configuration

Finally, Figure 3.10 shows the LBR distributing the traffic based on information carried in the Mitigation Label to two LSPs. The two LSPs are called priority and mitigation paths. The *priority* path carries the traffic that should be preserved from congestion as much as possible. The *mitigation* path carries traffic that may be dropped by the network if congestion occurs. In the remainder of chapter, we consider that iLSR and LBR functions are provided by the same equipment, located at the network ingress, and we call this equipment the LBR.

While most MPLS functions mentioned above (*e.g.* Traffic Engineering, Differentiated Service) are commonly implemented in routers, the proposed load-balancing mechanism requires additional router developments, as described later. Considering the mitigation application, we restrict the load-balancing to the sFEC, which is defined by a minimal length destination prefix that includes the victim IP addresses. As such the sFEC can be mapped to destination based route in the routing table.

We define the *Mitigation Label* as an extended special purpose MPLS label [110] pushed by the ingress LSR on the MPLS label stack of traffic. It carries information about suspicious flows taking part to an ongoing attack. We introduce in Subsection 3.4.2 the tBF, a network impact aware structure

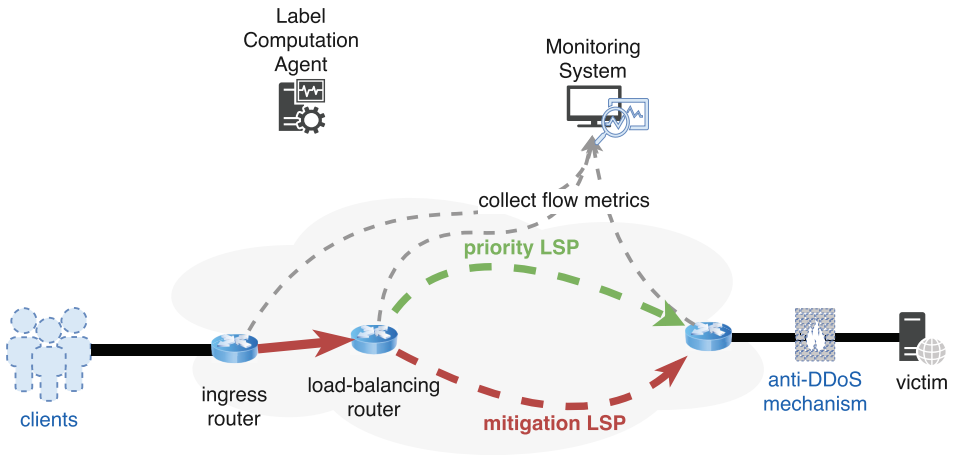


Figure 3.10 – Attack mitigation

that will store suspicious flows.

Volumetric DDoS Alert

Volumetric DDoS alerts are used by a detection mechanism to report that a volumetric DDoS attack has been detected, is still ongoing or has stopped. When the alert is related to a new or an ongoing attack, it contains the characterization of traffic elements reported as malicious, *i.e.* the identification of suspicious IP flow (IPf)s. The characterization of suspicious IP flows consists of a list of $<$ `source IP, destination IP` $>$ tuples associated with the attack. In fact, the configuration of the mitigation mechanism requires the description of suspicious IP flows, where the source and destination IPs respectively refer to the amplifier and target IP addresses. Defined this way, the IP flow includes only one direction of traffic and for example a TCP connection will result in two IP flows.

Relevant information exchange formats, for example intrusion detection IDMEF [144]) and DDoS attack signaling (from the IETF DDoS Open Threat Signaling Working Group [145]), carry such information.

Flow Telemetries

Our mitigation mechanism also requires flow telemetries, such as volumetries, related to the sFEC. We expect that both suspicious and legitimate volumetries are available. While metrics of suspicious flows may be carried by the alert message, we consider that flow metrics can also be available through network monitoring systems collecting telemetries using, for example, NetFlow [146], sFlow [147] or IPFIX [148]. In fact, if the system monitors traffic at least at the flow granularity, we should be able to collect volumes

for both suspicious and legitimate flows.

Traffic Updates

During an attack, both malicious and legitimate traffic vary over time. New flows may appear, and existing flows may terminate. The overall flow volumetry is changing. Hence, the mitigation label that depends on inbound traffic, has to be regularly recomputed to take into account the updated traffic data. The recomputation process is similar to the initial computation, with updated flow statistics. The minimum possible duration of the computation iteration is constrained by either the mitigation configuration, *i.e.* the label generation and advertisement, or the sampling period of the monitoring system. We found that the mitigation label computation lasts less than a second in our setting. If the mitigation and priority paths have already been established, the Mitigation Label may be advertised in the range of seconds. The monitoring system refresh period is likely larger and thus defines the shortest Mitigation Label update interval.

In the next section, we introduce the data structure required to achieve our load-balancing scheme.

3.4.2 Threshold-based Bloom Filter

Our solution is based on the capability of a router to deterministically load-balance traffic through a priority and mitigation path, *i.e.* enforce suspicious flows to be forwarded through the mitigation path while letting remaining flows through the legitimate path. We then rely on a lightweight signature of the attack, so that it can be embedded within the packet.

We beforehand study the usability of Bloom Filters to achieve our goal. Facing the impossibility of using BFs in our context, we introduce the Threshold-based Bloom Filter (tBF). In the following, we detail an intelligent generation of the tBF that take care of the network bandwidth allocation.

Usability of Bloom Filters

A Bloom Filter [41] (BF) is a space efficient data structure that represents the elements of a set S (*cf.* Table 3.2). It supports membership queries on S over a domain D, where responses only allow false positive, *i.e.* return an element x as a member of S, while it is not. The probability of false positive can be minimized by tuning three parameters [149]:

1. the size of the data structure, which is a m-bit array. The larger the size, the smaller the probability of a false positive is.

Parameter	Definition
S	set of elements that will be tested for membership
D	definition domain of set S
k	number of hash functions
m	BF length, *i.e.* number of bits in the bit array
n	number of elements of S to insert in the BF

Table 3.2 – Bloom Filter parameters definition

2. the number n of members in S. Reducing n leads to minimizing the probability of false positives. However, this criterion may not be always tunable as is can depend on the filter application. Besides, it is not always possible to know how many elements will be stored. Hence, there is a trade-off between over-dimensioning the filter and underestimating the false positive probability.

3. the number k of hash functions used both to generate and query the Bloom Filter.

Each bit of the Bloom filter array is initialized with 0. Assume that hash functions map elements of D to an index of the Bloom filter bits array. The Bloom filter is constructed by hashing each element of the set S (n elements in total) with the k hash functions. We then set the BF bit to 1 at position returned by the hash.

The membership test of an element is realized by hashing this element with the k hash functions. If any of the k^{th} array position given by the hash result is set to 0, the element is surely not a member of S. Otherwise, we declare with $1 - p_{FP}^{BF}$ probability that the requested element is a member of S, where p_{FP}^{BF} is defined in the Equation (3.6).

$$p_{FP}^{BF} = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k} \quad . \tag{3.6}$$

We use Equation (3.6) to assess the valid use of Bloom filters against a simple use case. We place ourselves in the case of a volumetric DDoS attack which involve 40,000 suspicious flows. Each flow has a valid source IP address, *i.e.* it had not been spoofed, for example considering an amplification DDoS attack. We then try to embed a Bloom Filter into the 20 bits Mitigation Label. Although defining a set S of 40.000 suspicious flows leads to a false positive probability of 1 regardless of the number of hash functions (k), *i.e.* $p_{FP}^{BF} = 1$. It means that testing a legitimate flow against the filter will give the flow as suspicious with a probability of 1, which is improper for traffic filtering. As a consequence, as each bit has a probability of 1 to be set to 1, the filter has surely all his bits set to 1. This is due to the restricted

length of the Filter (20 bits). According to the false positive Equation (3.6), the Bloom filter requires to have a length of at least 384.000 bits, with 6 hash functions. The number of hash function is obtained from the optimal space/time trade-off [41] given by Equation (3.7).

$$k = -\frac{m}{n * ln_2(FP)} \tag{3.7}$$

In order to address the high probability of false positives, we can reduce the number of suspicious flows by aggregating them. Although this solution may be simple, it restricts to 13 the maximal number of aggregates (*i.e.* n elements) that could be inserted into the filter to keep the trade-off (*cf.* Equation (3.7)) using only one hash function. In the next part, we present threshold-based Counting Bloom Filters, an extension to Bloom filters that allow us to enter larger number of flows into the filter.

Threshold Definition

As mentioned earlier, the size limitation of 20 bits imposed by the MPLS label renders plain BF inapplicable for us. We introduce tBF to address this issue. tBF is based on both a Bloom Filter, as defined in the previous section, and a Counting Bloom Filter (CBF) [150].

Counting Bloom filters use a counter array instead of the Bloom filter bit array. The filter is generated by incrementing array positions given by the hash results of the element to add as member instead of only setting it to 1. The membership query is the same bitwise operation as for Bloom filters.

We use a CBF of size m with a single hash function ($k = 1$) whose values are integers in the range $[0; m-1]$. Under these circumstances, an element, *i.e.* a flow, is described with only one counter. When we insert a flow into the CBF, the flow volume (in terms of bytes over the monitoring system refresh period) is added to the counter at position i, given by the flow hash result. The flow volumes have been previously retrieved from the monitoring system (Section 3.4.1). The total volume of flows whose hash gives the same result i, is then equal to the CBF counter value at position i, *i.e.* the i^{th} counter of the CBF. Considering furthermore the domain S of all flows inserted into the CBF, we define a subset S_i of S as the set of flows f of S that have the same hash result i.

We define the abstract concept of a major subset as a subset of malicious traffic which should be filtered due to the overall volume of the subset being important. The minimal volume which characterizes a major subset is defined by the threshold t, a positive integer. Each counter is compared to the threshold. If the counter is greater or equal to the threshold, the bit of the same position in the tBF is set to 1. Hence the tBF is a Bloom filter inferred from the CBF by applying a threshold to each counter (Figure 3.11).

A counter at a given position, whose value is below the threshold, leads to set the tBF bit at this position to 0. As we use tBF in the same manner as a BF, setting the i^{th} tBF bit to 0, implies that all flows whose hash result is i are considered as not members of S, which in turn induces false negatives for the membership queries made with tBF.

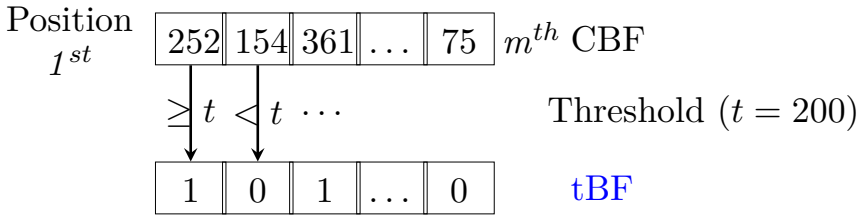


Figure 3.11 – Example of threshold-based Bloom filter generation given a threshold of $t = 200$

Constraint-based tBF Generation

This section details the computation of the tBF which aims at preventing the priority link saturation. We use the following definitions:

S The set of elements to be inserted in the tBF (suspicious flows in our case).

D The set of elements that can be tested against the tBF (all possible flows in our case).

L The set of most frequently tested elements that are not in the set S (the most active legitimate flows in our case).

The suspicious flow set (S), retrieved from DDoS alerts received by the Monitoring System, is used to generate the tBF and select a threshold. If elements of L, *i.e.* frequent legitimate flows, are known, they are also used to compute the threshold. Otherwise L is an empty set ($L = \emptyset$). To rank subsets as major or not, we sort them by taking into account both legitimate and suspicious traffic. First, we insert legitimate and suspicious flows into their own CBF with a size of $m = 20$, respectively CBF_L and CBF_S. Second, we introduce another 20-bit CBF (CBF_{mixed}) to carry the impact of the volume of subsets:

$$CBF_{mixed}[i] = \frac{CBF_S[i]}{1 + CBF_L[i]}, \forall i \in [0; m-1] \ .$$

This initialization:

1. increases the weight of the mixed counter i along with $CBF_S[i]$ counter (representing the volume of S_i),

2. decreases the weight of the mixed counter i along with $CBF_L[i]$ counter, and

3. allows the case where the set L is empty to be assessed.

Considering a given tBF, we define the volume of false positives (V_{FP}) as the sum of volumes of flows in subsets L_i given by the position i such that the i^{th} bit of the tBF is set to 1. Conversely, the volume of false negatives (V_{FN}) is equal to the sum of CBF_S counters given by the position $\bar{i}$ such that the $\bar{i}^{th}$ bit of the tBF is null, *i.e.*:

$$V_{FP} = \sum_{i} CBF_L[i], \text{ such that } tBF[i] = 1 \ ,$$
$$V_{FN} = \sum_{\bar{i}} CBF_S[\bar{i}], \text{ such that } tBF[\bar{i}] = 0 \ .$$

The generation of the tBF is a constraint-based process. We intend to prevent the priority link, which has allocation A_{priority}, from being saturated. We estimate the volume of the traffic going through the priority link as the sum of volumes for false and true negatives. Given that the volume of true negatives is equal to the difference between the incoming legitimate traffic volume (expressed by the $V_{in}leg$ variable) and V_{FP}, we describe the following constraint:

$$V_{in}leg - V_{FP} + V_{FN} < A_{\text{priority}} \ . \tag{3.8}$$

We first set the threshold to its maximal value (*i.e.* the sum of all mixed counters) and decrease it until the constraint (*cf.*, Equation (3.8)) is satisfied. However, to avoid iterating the constraint check with all possible thresholds, we sort subsets S_i of S by their impact, *i.e.* we sort CBF_{mixed} counters by the decreasing order. We then initialize the tBF with 0s and one by one set the bit at position i to 1, where positions are sorted according to their impact. We stop setting bits to 1 when the constraint on the priority link allocation is satisfied (Equation (3.8)). If all tBF bits have been set to 1 before the constraint is satisfied, we reset to 0 the tBF bit at position i, such that S_i has the smallest impact. In this way, we prevent all inbound traffic to be load-balanced through the mitigation path.

3.4.3 Mitigation Label-based Load-Balancing

Based on the observation that probabilities of false negatives and positives have opposite trends in function of the threshold t (Equation (3.15)), we embed a twofold filter in the Mitigation Label (*cf.* Figure 3.12). The Mitigation Label consists of a BF into which attack targets have been inserted and a tBF containing suspicious IP flows reported by the monitoring system. We denote as TSL length of the Bloom filter. The size of the tBF is then $20 - TSL$.

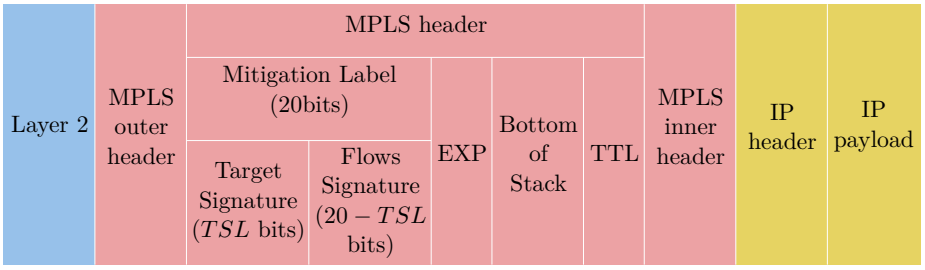


Figure 3.12 – MPLS header with Mitigation Label

As show in Figure 3.13, once a packet belonging to the sFEC hits a LBR, it reads the Mitigation Label as well as flow details needed to compute the flow hash, *i.e.* source and destination IPs. Then the LBR compares the hash of the destination IP with the signature of targets obtained from the TSL first bits of the Mitigation Label. A packet which destination IP would not be recognized as belonging to targets according to this signature is directly forwarded through the priority LSP. Otherwise, the LBR test the hash of source and destination IPs against the signature of flows obtained from the remaining bits of the Mitigation Label. If the packet matches the signature, the LBR forwards it through the mitigation LSP, otherwise, it uses the priority LSP.

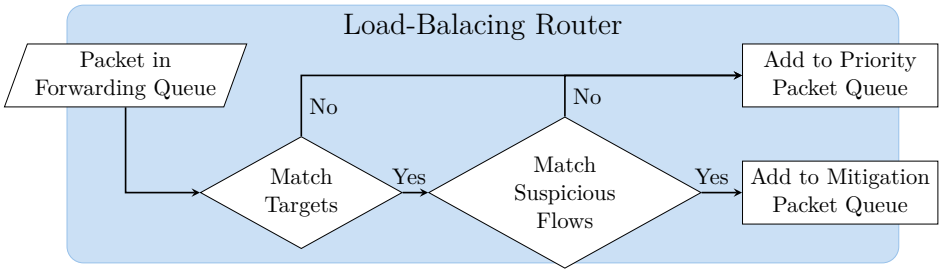


Figure 3.13 – Twofold Filter Flowchart

Targets Signature One of the basic characteristics of a DDoS attack is that, source IPs are much more numerous than the number of targets (few destination IPs). As suspicious flows refer to amplifier and target IPs pairs, the number of suspicious flows may be even superior to the number of amplifiers. In order to reduce the number of false positives, a packet is first matched according to its destination IP address against a signature of targets, *i.e.* a Bloom Filter containing target IPs. The signature of the target is extracted from the first bits of the Mitigation Label.

Suspicious Flow Signature As shown in Figure 3.13, packets matching the targets signature are tested against a signature of suspicious flows. This signature is a tBF into which hashes of suspicious flows have been inserted using the hash function H.

$$H(S, D) = ((s_1 s_2) \oplus (s_3 s_4) \oplus (d_1 d_2) \oplus (d_3 d_4)) mod\ N \tag{3.9}$$

where s_i and d_i are respectively the i^{th} bytes of the source IP S and destination IP D.

We have seen in Section 3.3 that using IP header in MPLS forwarding is subject to protocol misinterpreting and causes an extra burden. Since the sFEC contains only IP traffic, the induced overhead is limited. In fact, no guess on the inner payload protocol is required, such that when the load-balancing router detects a Mitigation Label, it knows for certain that the inner payload is the IP protocol. This eliminates the source of misinterpretation. For example, this can be enforced at the ingress LSR by defining the sFEC as an IP destination prefix, such that only IP traffic is mapped to the sFEC.

3.4.4 Security of the Mitigation Label

Assuming that the core network is secure, *i.e.* no attacker has gained administrator rights on MPLS domain routers or can inject traffic from within the MPLS domain, we discuss the security of Mitigation Label.

The threat is the ability for an attacker to spoof the Mitigation Label so that attack traffic can evade the mitigation path. As we assumed that the attacker has not compromised the MPLS domain, and since the Mitigation Label is imposed by the LER and pushed onto the label stack, the Mitigation Label cannot be altered. Additionally, regarding amplification DDoS, attacker does not control the amplifiers. He can only affect the Application payload of the amplifier response by forging the application request. Hence from a victim point of view, malicious traffic is healthy for the Mitigation Label.

Considering that the attacker is able to forge the whole datagram sent towards the target, *e.g.* botnet-based attacks, he can then forge an already labelled packet [151]. If the incoming spoofed label is invalid, or the LER does not accept incoming MPLS encapsulated packet, the forged traffic is dropped [110]. Since the load-balancing forwards an initial FEC (the sFEC) to a priority and mitigation LSP, a valid spoofed label will cause the packet to no longer be recognized as belonging to the sFEC and it will not be forwarded to the desired victim.

3.5 Probabilistic Evaluation of Mitigation Label

tBF stores a large number of suspicious flows in a very small bit array (20 bits). tBF suffers from false positive errors (FP), as for Bloom Filters. It also suffers from false negative results (FN), *i.e.* wrongly returns that an element is not stored in the filter. In Subsection 3.5.1 we then study the probability of both false negative and false positive in order to validate the use of the tBF and find a trade-off between FN and FP. As it is difficult to find such a trade-off, we study in Subsection 3.5.2 the probability of false and positive of a twofold load-balancing process that will first use the destination IP and second both source and destination IPs.

3.5.1 tBF False Positive and Negative Probabilities

By construction of the tBF (*cf.* Figure 3.11), an element is considered as a member of S (the set of elements inserted in the tBF) only if the bit at the position (i) given by its hash ($hash(element) = i$) in the inferred BF (tBF) is set to 1. This means that the value of the CBF counter given by the position i ($CBF[i]$) determines the query result. If the counter is greater or equal to the chosen threshold t, *i.e.* $CBF[i] \geq t$, the element is returned as member.

As for Bloom filters, we assume that hash functions uniformly distribute flows to counters, *i.e.* each counter has equal probability to be chosen. Hence considering the insertion of an element in a y-size tBF, the probability that a counter at a given position is incremented is:

$$p(cnt = 1) = \frac{1}{y}. \tag{3.10}$$

Since we insert n elements, the probability that the value of a counter is strictly inferior to a threshold t is given by Equation (3.11), according to the cumulative distribution function of the binomial distribution.

$$\begin{aligned} p(counter < t) &= p(counter \leq t-1) \\ &= \sum_{k=0}^{t-1} \binom{n}{k} p(counter = i)^k (1 - p(counter = i))^{n-k} \end{aligned} \tag{3.11}$$

An element is considered as member of the set S (positive) if its hash is mapped to a counter that exceeds the threshold t. The result only depends on the counter value, and not on the fact that the element is a member of S or not. In other words, the fact that an element is a member of S does not affect the result, only the hash result matters. The test result of an element is independent of the fact that it is a part or not of the set S. Due to this independence, the probabilities that an element is a false positive or negative are respectively equal to the probabilities of positive and negative

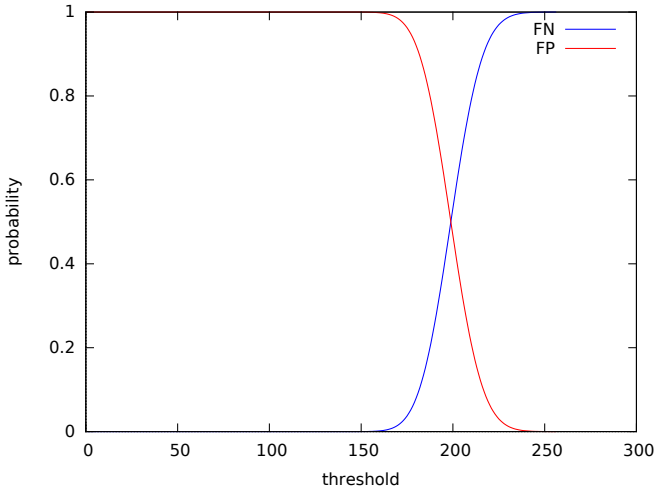


Figure 3.14 – Probabilities of Simple tBF Load-Balancing considering the insertion of 4096 flows into a 20bits tBF

result (*cf.*, Equation (3.13)). It does not depends on the fact that the element is a member of S.

$$p_{FN}^{tBF} = p_{negative}^{tBF} \tag{3.12}$$

$$p_{FP}^{tBF} = p_{positive}^{tBF} \tag{3.13}$$

where,

$$p_{negative}^{tBF} = p(counter < t) \tag{3.14}$$

$$p_{positive}^{tBF} = 1 - p_{negative}^{tBF} \tag{3.15}$$

In a given context (*i.e.* $n = 4096$ and $y = 20$), Figure 3.14 shows the impact of threshold on the probability of false positive and negative (p_{FP}^{tBF} and p_{FN}^{tBF}).

Given this context, the probability of false positive would always result to 1 for a Bloom Filter. However, p_{FP}^{tBF} and p_{FN}^{tBF} curves show opposite trends, as expressed in Equation (3.15). The threshold for which the probability of false negative and positive cross, is the theoretical trade-off between collateral damages and filtered attack traffic. In the context of volumetric DDoS attacks, we should however preferably maximize the volume of dropped attack traffic to relieve network traffic congestion. Once the DDoS condition is eliminated, we should try to minimize collateral damages.

Curves given by the probabilities are our worst case, where counters of the filter have equal probabilities to be chosen. In real case, the distribution of malicious flows might not be equally distributed among counters. Hence counters with higher value than the one given by probability may have more weight on threshold choice.

3.5.2 Two Steps Load-Balancing Probabilities

As tBF evenly treat flows destined or not destined to a target, we introduced the Target Signature in Subsection 3.4.3. The number of false positive errors is reduced for IP flows that are not destined to a target, without probabilistically impacting the number of false negative errors. In this section, we study the theoretical impact of the target BF by assessing the probability a flow being forwarded through the priority or mitigation LSP. Figure 3.15 shows the conditional probabilities on the event associated with the match by both filters.

Probability of False Negatives We express the probability of negatives, *i.e.* a flow to be forwarded through the priority path, as following:

$$P^{ML}_{negative} = P(\overline{tBF} \cap (BF \cap T) + P(\overline{BF} \cap T) + P(\overline{tBF} \cap (BF \cap \overline{T}) + P(\overline{BF} \cap \overline{T})$$

Using the Kolmogorov definition of conditional probabilities, we expressed the probability of a negative event with the probabilities of both BF and tBF.

$$\begin{aligned} P^{ML}_{negative} = {} & P(T) \times \left(P(BF|T) \times P(\overline{tBF}) + P(\overline{BF}|T)\right) \\ & + P(\overline{T}) \times \left(P(BF|\overline{T}) \times P(\overline{tBF}) + P(\overline{BF}|\overline{T})\right) \end{aligned}$$

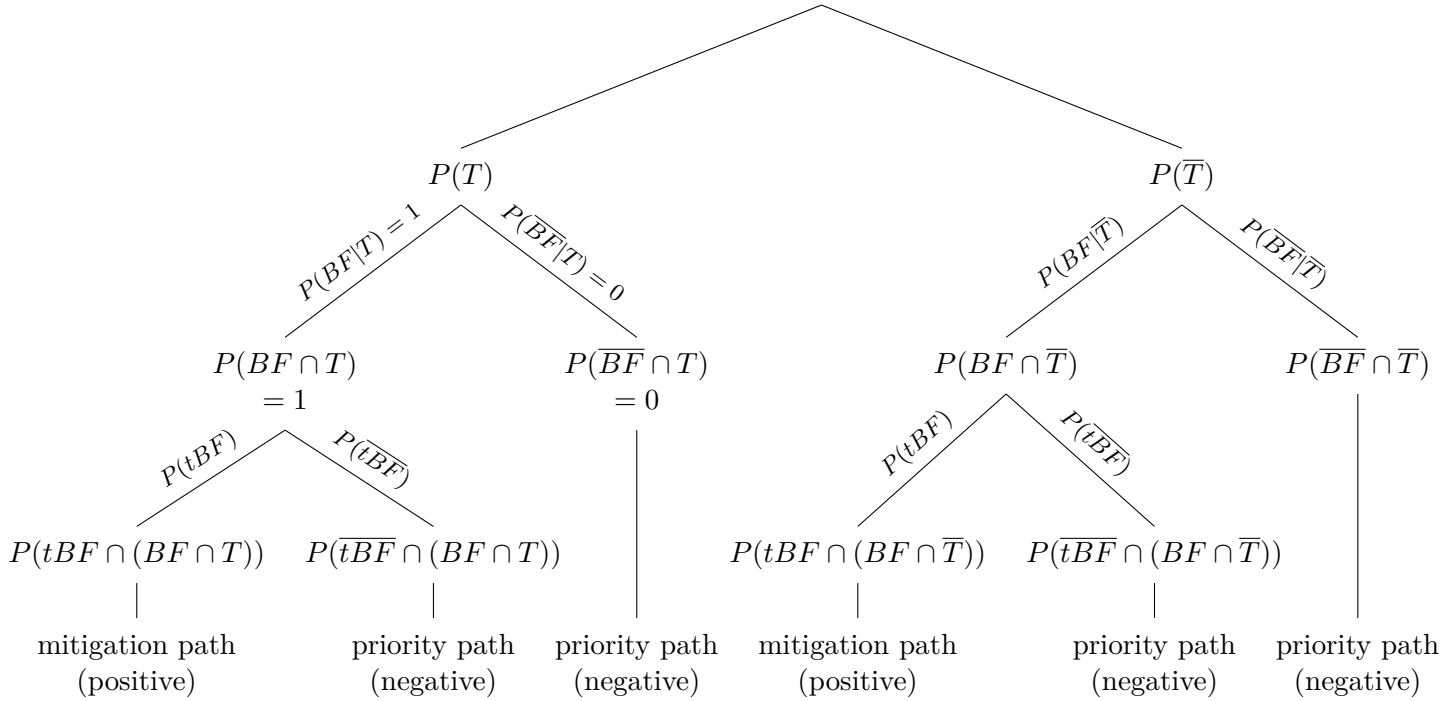
We know that BFs do not induce false negative errors, *i.e.* $P(\overline{BF}|T) = 0$ and $P(BF|T) = 1$.

$$\begin{aligned} P^{ML}_{negative} = {} & P(T) \times P(\overline{tBF}) \\ & + P(\overline{T}) \times \left(P(BF|\overline{T}) \times P(\overline{tBF}) + P(\overline{BF}|\overline{T})\right) \end{aligned}$$

Considering the probability of false negatives, we only examine suspicious flows. As a consequence, the probability that the destination IP of the flow is not a target is null ($P(\overline{T_S}) = 0$).

The probability of false negatives can then be expressed as :

$$\begin{aligned} P^{ML}_{FN} = {} & \underbrace{P(T_S)}_{=1} \times P(\overline{tBF}) \\ & + \underbrace{P(\overline{T_S})}_{=0} \times \left(P(BF|\overline{T_S}) \times P(\overline{tBF}) + P(\overline{BF}|\overline{T_S})\right) \\ = {} & P(\overline{tBF}) \\ = {} & p^{tBF}_{negative} \end{aligned} \tag{3.16}$$



T: destination IP of flow is a volumetric DDoS attack target
BF: destination IP of flow matches the target signature (*i.e.* Bloom Filter return positive result)
tBF: flow hash matches the suspicious flow signature (*i.e.* threshold-based Bloom Filter return positive result)

Figure 3.15 – Decision tree of the Mitigation Label conditional probabilities

Probability of False Positives The probability that a flow will be forwarded through the mitigation path (positive result probability) is given by the following equation (*cf.* Figure 3.15):

$$P^{ML}_{positive} = P(tBF \cap (BF \cap T)) + P(tBF \cap (BF \cap \overline{T}))$$

The probability of false positives can then be expressed using the probability of a positive event of both BF and tBF :

$$\begin{aligned} P^{ML}_{FP} &= P(T) \times \underbrace{P(BF|T)}_{=1} \times P(tBF) \\ &\quad + P(\overline{T}) \times P(BF|\overline{T}) \times P(tBF) \\ &= P(T) \times p^{tBF}_{positive} + (1 - P(T)) \times p^{BF}_{FP} \times p^{tBF}_{positive} \end{aligned} \tag{3.17}$$

However, it is difficult to evaluate the probability of a legitimate flow to have its destination among the targets (*i.e.* $P(T_L)$). It depends on the number of legitimate flows flowing towards the targets in the sFEC ($\#legIPf(dest = targets)$) and the total number of legitimate flows ($\#legIPf$).

$$P(T_L) = \frac{\#legIPf(dest = targets)}{\#legIPf} \tag{3.18}$$

However, the probability of false positive is bounded by $p^{tBF}_{positive}$ and $p^{BF}_{FP} \times p^{tBF}_{positive}$, as proved thereafter.

Let a, x and y respectively represent $P(T_L)$, $p^{BF}_{FP} \times p^{tBF}_{positive}$ and $p^{tBF}_{positive}$. We get the following assertions:

$$\begin{cases} 0 \leq a \leq 1, & \text{as } a \text{ refers to a probability} \\ x \leq y & \text{as } 0 \leq p^{BF}_{FP} \leq 1 \ . \end{cases} \tag{3.19}$$

Then :

$$\begin{aligned} x \leq y &\iff ax \leq ay && \text{as } 0 \leq a \\ &\iff 0 \leq ay - ax \\ &\iff x \leq ay + (1 - a)x \\ &\iff p^{BF}_{FP} \times p^{tBF}_{positive} \leq P(T) \times p^{tBF}_{positive} + (1 - P(T)) \times p^{BF}_{FP} \times p^{tBF}_{positive} \end{aligned}$$

And we also get:

$$\begin{aligned} x \leq y &\iff (1 - a)x \leq (1 - a)y && \text{as } 0 \leq 1 - a \\ &\iff ay + (1 - a)x \leq y \\ &\iff P(T) \times p^{tBF}_{positive} + (1 - P(T)) \times p^{BF}_{FP} \times p^{tBF}_{positive} \leq p^{tBF}_{positive} \end{aligned}$$

3.5.3 Evaluation

Figure 3.16 shows the probability of false negatives (line with points) of a twofold Mitigation Label. The probability of false positive given that the flow destination is or is not a target ($P^{ML}_{FP|T}$ and $P^{ML}_{FP|\overline{T}}$) are respectively depicted in squares and triangles. Finally, the curve with plus signs illustrates the probability of false positive given that 75% of legitimate flows are destined to a target.

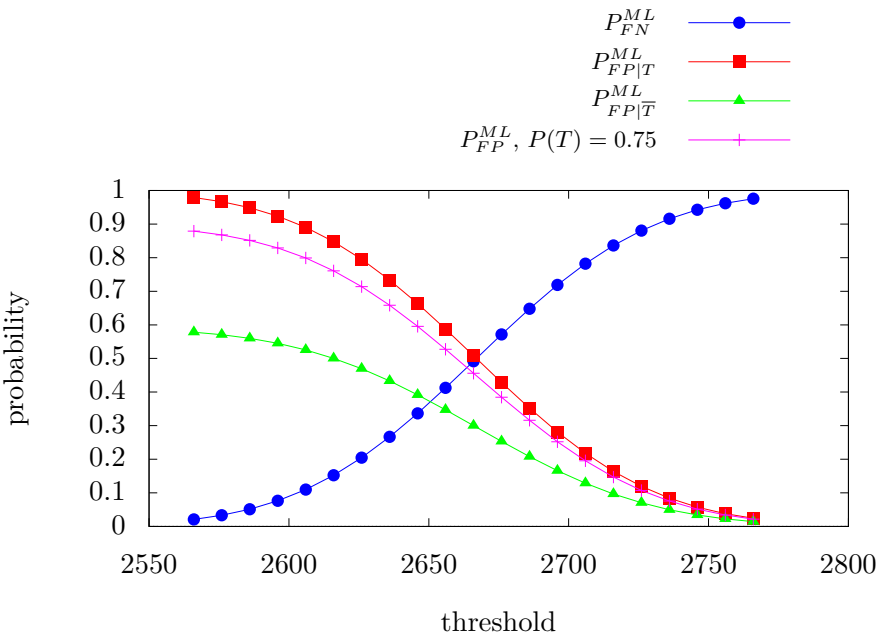
Considering a given Target Signature Length (TSL), the probability of false positives given that the flow destination is or is not a target (lines with squares and triangles) show the same decreasing trends. In fact, a low threshold induces that potentially more counters would be above the threshold. As a consequence, more legitimate flows would then be filtered as they are mapped to these counters. Moreover, both lines with squares and triangles get closer, as the threshold grows. This is due to the dependence of $P^{ML}_{FP|\overline{T}}$ on $P^{ML}_{FP|T}$ (*cf.*, Figure 3.15).

The potential gain of the destination-based pre-filter is highlighted by the difference between the lines with squares and triangles, *i.e.*, probabilities of false positives given that the destination is or not a target. Using a longer TSL increases this potential gain. In fact, the probability of false positive given that the destination is not a target is equal to 0.6 and 0.4 for a TSL of respectively 5 and 8 (given a threshold value of 2575). However, considering a given proportion of legitimate flows destined to the targets, the gain is reduced. For example, given that 75% of legitimate flows are flowing towards a target, the false positive probability is equal to 0.9 and 0.82 for a TSL of respectively 5 and 8 (threshold = 2575).

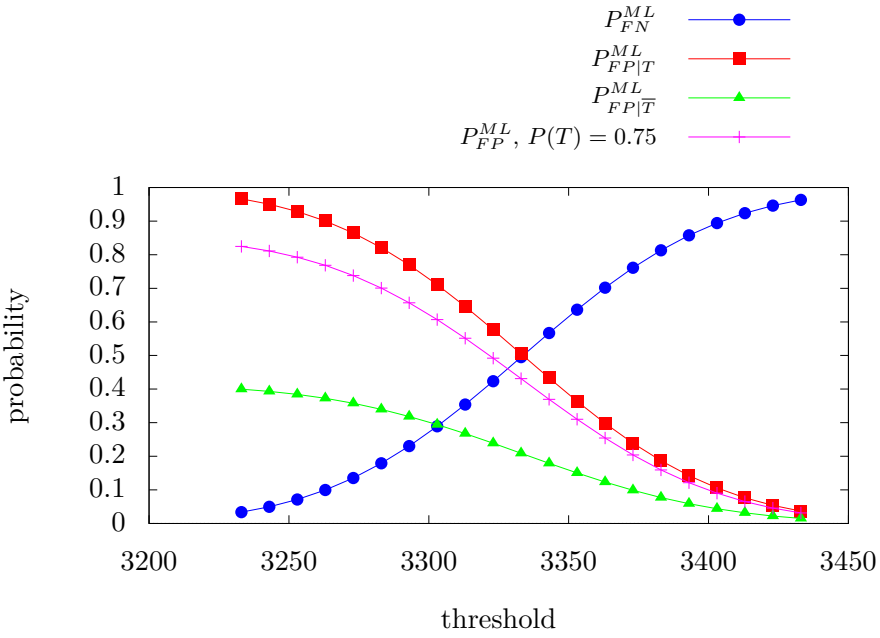
Evaluation shows that a smaller threshold probabilistically induces less false positives. Nonetheless this also results in increasing the probability of false negatives, which may lead to forward enough flows to saturate the network. In fact, while this evaluation highlights the balance between required probabilities of false positives and negatives, it does not carry the congestion condition, *i.e.* whether a given threshold allows preventing the link saturation.

3.6 Experiments

This section aims at validating the use of the Mitigation Label to mitigate volumetric DDoS attacks while minimizing collateral damages. We conduct experiments by replaying traffic captures against a software implementation of our load-balancing mechanism. The experimental approach is described in Subsection 3.6.1. Then Subsection 3.6.2 details variables and metrics used to study the behavior of our mechanisms under different angles. Subsection 3.6.3 depicts the impact of the Target Signature Length (TSL) on the



(a) $TSL = 5$



(b) $TSL = 8$

Figure 3.16 – Probability of false positives and negatives of Mitigation Label

mitigation results. Subsection 3.6.4 studies the behavior of our mechanism according to different network setup.

3.6.1 Experimental approach

Our mitigation mechanism has been implemented as a software platform. The platform takes as input traffic captures representing the network ingress traffic and outputs two captures representing both priority and mitigation links. The platform load-balances input traffic through a priority and a mitigation buffer. Subsequently, buffers are throttled using a token bucket algorithm [84] to enforce the bandwidth constraints of priority and mitigation output captures (representing links).

Experiments are conducted using a 5 second sliding window: 1) both legitimate and suspicious traffic telemetries are gathered from traffic capture during 5 seconds, 2) the mitigation label is then computed and applied to the next 10 seconds, 3) at the 5th second of the second step, we loop into the first step. For evaluation purposes, metrics (Sect. 3.6.2) are extracted from the mean of measures obtained from each iteration.

We generate input traffic captures by mixing legitimate traffic captures extracted from the MAWI data set [152] and several generated attack captures. We select /24 destination-based aggregates (cf. the sFEC) from MAWI captures as legitimate traffic. The average legitimate traffic bandwidth of each capture is between 14 and 16 Mbits per second. Attack traffic is generated by replaying DNS ANY replies from 40000 randomly chosen source IPs (amplifiers) towards 4 targets selected in the sFEC destination IPs. In fact, although Rossow [11] found up to billions of potential amplifiers, the attack with the highest number of amplifiers reported only involved around 17000 different IP sources.

3.6.2 Variables and Metrics

Figure 3.17 shows measures used to conduct the evaluation, and their point of collection. We first measure inbound legitimate (green hashed) and malicious (red filled) traffic volumes, respectively named $V_{\text{in}}\text{leg}$ and $V_{\text{in}}\text{mal}$. The traffic is load-balanced between priority and mitigation LSPs, which have allocation A_{priority} (Sect. 3.3.1) and $A_{\text{mitigation}}$ bandwidths, respectively. $V_{\text{priority}}\text{leg}$ denotes the volume of legitimate traffic forwarded through the priority link. Finally, $V_{\text{out}}\text{leg}$ refers to the outgoing legitimate traffic volume that has been forwarded either through both paths, and that has not been dropped by the QoS mechanisms.

Our solution aims to maintain an acceptable level of Quality of Service for legitimate traffic. In the experiments we simply assess the QoS through its availability aspect by measuring traffic losses. The evaluation does not cover more complex aspects such as delay and jitter.

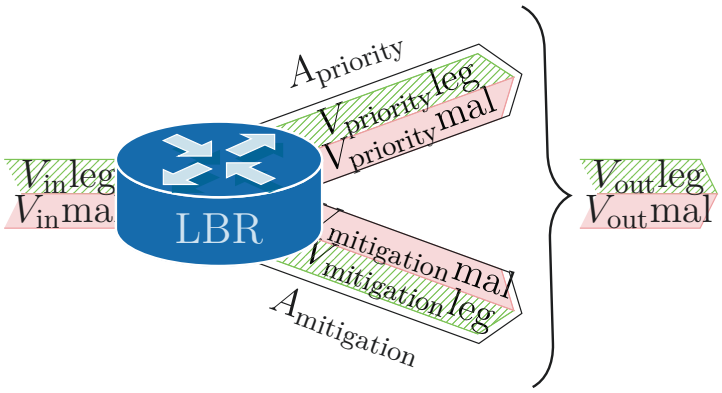


Figure 3.17 – Load-Balancing Measures

Metric	Equation	Definition
Reception	$\frac{V_{\text{out}}\text{leg}}{V_{\text{in}}\text{leg}}$	Reception of legitimate traffic
$R_{priority}$	$\frac{V_{\text{priority}}\text{leg}}{V_{\text{in}}\text{leg}}$	Reception of legitimate traffic on the priority LSP

Table 3.3 – Definition of evaluation metrics

Metrics The efficiency is evaluated with a metric we call *Reception* (*cf.* Table 3.3) which is the ratio between the outbound and inbound legitimate volumes. It evaluates how much legitimate volume our solution is able to preserve.

We also define the Reception on priority LSP ($R_{priority}$) as the ratio of the volume of legitimate traffic forwarded through the throttled legitimate path and the inbound legitimate traffic volume. This metric identifies the capability of our solution to effectively segregate malicious from legitimate traffic.

Variables The volume of attack traffic is controlled by the variable *AttackFactor* which is the ratio of volumes of inbound suspicious and legitimate traffic, $\frac{V_{\text{in}}\text{mal}}{V_{\text{in}}\text{leg}}$. As we deal with massive DDoS attacks, we generate malicious traffic with an attack factor ranging from 1 to 50. Large attack factors allow us to assess the limitations of our solution.

The theoretical usage of the priority path by the legitimate traffic impacts the efficiency (*Reception* metric) of our proposal. First, without considering attack traffic, legitimate traffic can, on its own, saturate the priority path. Second, given an allocation of priority path (A_{priority}), the more a customer uses its reserved bandwidth, the less malicious traffic is required to overwhelm the path. In order to model this traffic characteristic, we define the variable $Usage_{\text{priority}}$ as the theoretical (*i.e.* without allocation of mitigation path, $A_{\text{mitigation}} = 0$) percentage of utilization of the priority path.

Variable	Equation	Definition
$AttackFactor$	$\frac{V_{\text{in}}\text{mal}}{V_{\text{in}}\text{leg}}$	Ratio between ingress suspicious and legitimate traffic volumes
$Usage_{\text{priority}}$	$\frac{V_{\text{in}}\text{leg}}{A_{\text{priority}}} \times 100$	Theoretical utilization of the priority path by the legitimate traffic
$BA_{\text{mitigation}}$	$\frac{A_{\text{mitigation}}}{A_{\text{priority}}+A_{\text{mitigation}}} \times 100$	Mitigation LSP bandwidth allocation as a percentage of the overall bandwidth allocated to the customer
TSL		Target Signature Length (*cf.* Subsection 3.4.3

Table 3.4 – Definition of variables

$BA_{\text{mitigation}}$	$Usage_{\text{priority}}$
10%	55.6%
40%	83.3%

Table 3.5 – $Usage_{\text{priority}}$ values dictated by MAWI captures in function of the $BA_{\text{mitigation}}$

Considering that the volume of legitimate traffic is dictated by the MAWI captures, the evaluated values of $Usage_{\text{priority}}$ are shown in Table 3.5.

As discussed in Sect. 3.3.1, we have to find a tradeoff between the allocation for mitigation path and the mitigation efficiency, *i.e.* the value of *Reception* of legitimate traffic. From a network provider's perspective, we express the need for mitigation allocation per client as a percentage of the overall bandwidth allocated to the customer, *i.e.*$BA_{\text{mitigation}}$ in Table 3.4. We have considered two cases: 1) the overall bandwidth allocation ($A_{\text{priority}} + A_{\text{mitigation}}$) is fixed, *e.g.* the ISP has already provided an allocation for the mitigation path, 2) only the bandwidth allocation of the priority path is fixed, *e.g.* the mitigation path is provisioned during the mitigation. The cost of mitigation grows with the bandwidth allocated to the mitigation path. We then should find the optimum percentage of mitigation allocation that minimizes the mitigation allocation and maximizes the value of *Reception* of legitimate traffic.

3.6.3 Results on the impact of TSL

This part assesses how the Mitigation Label-based load-balancing is able to protect the legitimate traffic and forward it through the priority path, ac-

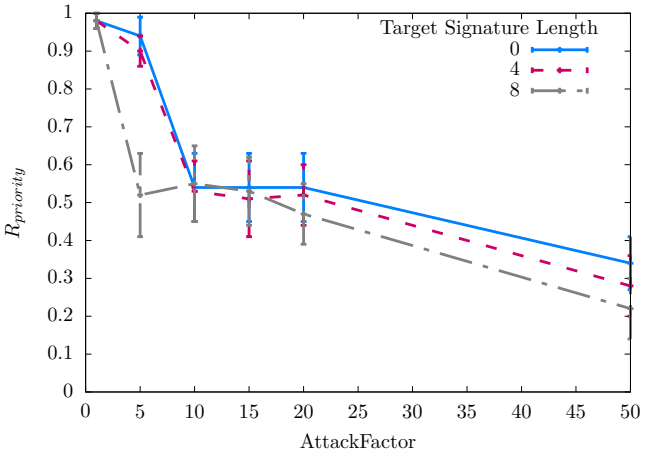
cording to the length of the target-based signature (TSL). We then only consider the priority path and its related variables ($Usage_{\text{priority}}$, TSL) and metrics (R_{priority}). Figures 3.18a and 3.18b show the impact of the $AttackFactor$ (defined in Subsection 3.6.2) on the reception of legitimate traffic on the priority path (R_{priority}). For each sub-figure, the $AttackFactor$ varies while the legitimate traffic volume remains the same. Curves in blue, red and grey respectively depict a TSL of 0 (*i.e.* no target signature), 4 and 8.

Considering a legitimate traffic that fulfills 55% of the priority allocation, Figure 3.18a shows that decreasing the target signature length (TSL) leads to a growth of the R_{priority} for a given attack volume. For example, the three curves (for the three considered TSL) have the same R_{priority} for attack which volume is equal to the legitimate traffic volume. Then the percentage of reception on the priority link given a TSL of 8 falls to 52% for an $AttackFactor$ of 5, while it declines more slowly to around 90% for a TSL of 0 or 4. The first fall of the R_{priority} metric is due to the fact that :

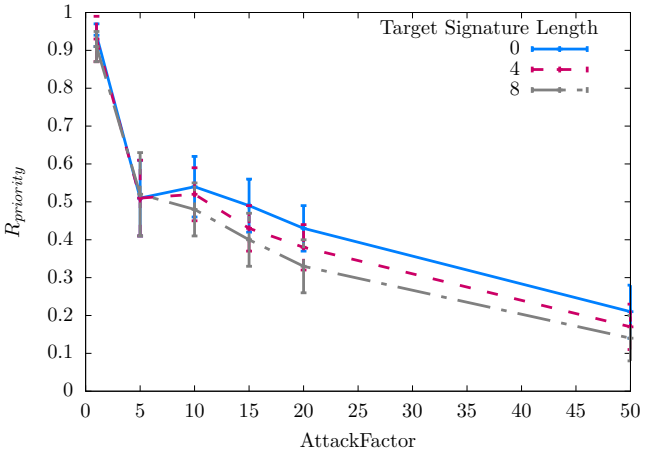
1. considering an incoming attack traffic which volume is equal to the legitimate volume ($AttackFactor = 1$), the overall incoming traffic bandwidth is equal to the overall output allocation ($mam + map$). In such case, the tBF is able to load-balance almost all of the legitimate traffic through the priority path on which no throttling is applied.

2. When the attack volume ($AttackFactor$) increases, more tBF bits are set to 1 in order to satisfy the bandwidth constraint of the priority link. In that case, the number of legitimate flows load-balanced to the mitigation path increases and the reception of legitimate traffic through the priority path decreases.

The same R_{priority} decline is delayed at an $AttackFactor$ of 10 for TSL 0 and 4. The delay is due to the fact that smaller tBF (*i.e.* larger TSL, *e.g.* 8-bit size) a bit (or counter) is the corresponding hash of a higher number of flows (including legitimate flows). Thus, at a given attack volume, less legitimate flows are load-balanced to the mitigation path with a TSL of 0 or 4 rather than of an 8-bit tBF. We begin to force a bit of the tBF to one due to a non-compliant priority bandwidth constraint from an $AttackFactor$ of 20 considering a TLS of 8 and 15 for TSL 4 and 0. The priority path is then throttled at a small attack volume for higher TSL. Figure also shows, specially curves representing a TSL of 4, that the R_{priority} increases while the attack volume increases. This is due to the characteristics of attack traffic. In fact, increasing the attack factor does not implies each flow volume to rise at the same rate as the number of malicious flows is important and $AttackFactor$ increment are small. The CBF of malicious flows may then be different considering two continuous $AttackFactor$. The standard deviation of $Reception$ remains rather high with an average top value of respectively



(a) 55% of the priority link used by legitimate traffic ($Usage_{\text{priority}}$)



(b) 83% of the priority link used by legitimate traffic ($Usage_{\text{priority}}$)

Figure 3.18 – Impact of the Target Signature Size on the reception of legitimate traffic on the priority link ($R_{priority}$) according to the attack volume expressed as a factor of the legitimate traffic volume ($AttackFactor$)

0.08 and 0.11. The dependency of the tBF on the flow entropy (distribution over hash values) explain this behavior. We conducted experiments with 6 different traffic captures.

The curves follow the same trends given a higher priority path utilization by the legitimate traffic ($Usage_{\text{priority}} = 83\%$, Figure 3.18b), *i.e.* the reception on priority link increases as the target signature length decreases (*i.e.* the tBF size increases). Also, as expected, for a given target signature length and attack volume, the reception of legitimate traffic on priority path is lower considering larger priority utilization by legitimate traffic.

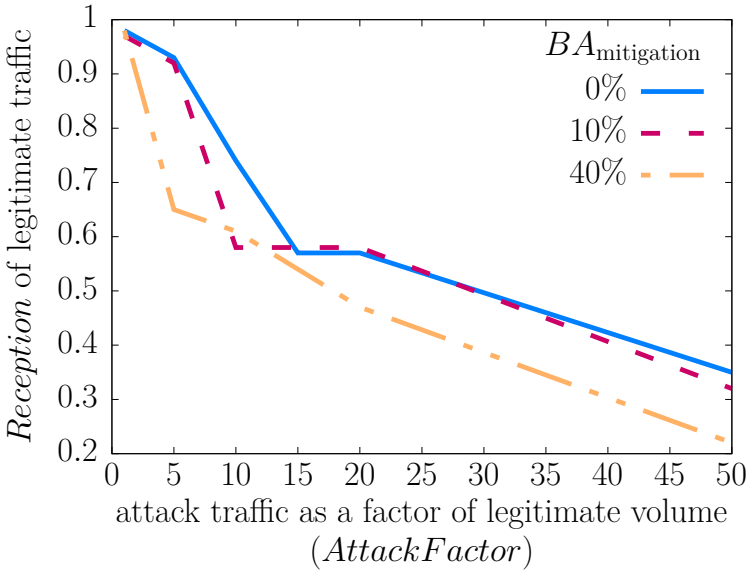
3.6.4 Results on bandwidth allocation

Following previous results, we fixed the TSL to 0, so that we do not use the target signature in the Mitigation Label. Figures 3.19a and 3.19b show the impact of the bandwidth allocated to the mitigation path on the reception of legitimate traffic.

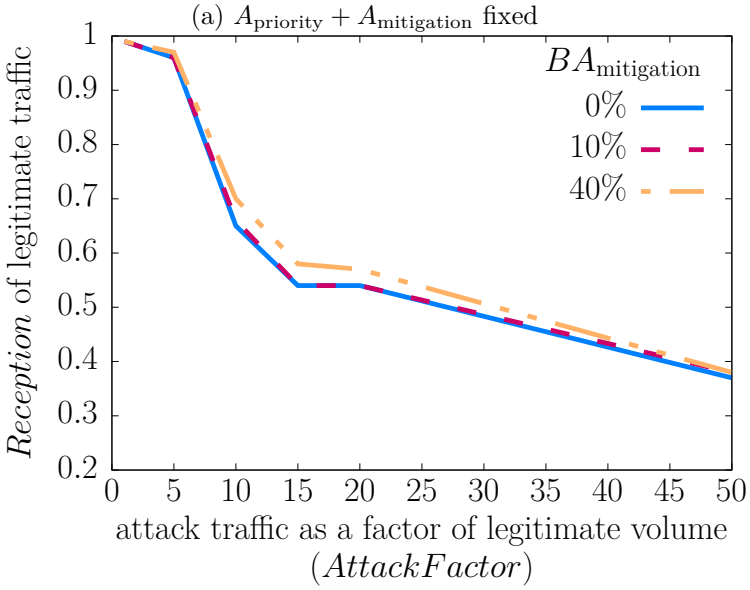
Figure 3.19a depicts variations of the bandwidth allocated to mitigation path considering a fixed overall bandwidth allocation ($A_{\text{priority}} + A_{\text{mitigation}}$). Regardless of the amount of the mitigation path bandwidth being allocated, the mitigation mechanism prevents legitimate traffic from suffering from congestion when the attack and legitimate traffic volumes are equal ($AttackFactor = 1$) and guarantees at least 90% of the legitimate traffic to be forwarded when the attack volume reach 5 times the legitimate volume with a $BA_{\text{mitigation}}$ strictly lower than 40%. Then the *Reception* drastically drops when the attack traffic reaches 5 times the legitimate volume considering a percentage of mitigation path allocation of 40%. The drop is shifted for a $BA_{\text{mitigation}}$ of 10% to an $AttackFactor$ of 10 and to an $AttackFactor$ of 15 when no bandwidth is allocated to the mitigation path. Indeed, considering larger allocations of mitigation link, more tBF bits are set to 1 to satisfy the priority bandwidth constraint leading more legitimate flows to be load-balanced to the mitigation path. Additionally, the mitigation bandwidth throttling is active from an attack volume that is greater than the legitimate traffic volume. From an attack factor of 20 to 50, the reception of legitimate traffic linearly declines from around 55% to 35% and 22% for a ratio of mitigation path allocation of 0 and 40% respectively.

In conclusion, allocating bandwidth to the mitigation path does not enhance the reception of legitimate traffic at the network egress. This is due to the fact that allocating bandwidth to the mitigation path while the overall allocation remains constant implies the reduction of priority path bandwidth allocation. Hence, in order to satisfying priority bandwidth constraints, the number of tBF bits set to 1 is increasing. That implies that more legitimate flows are load-balanced to the mitigation link which is more aggressively throttled than the priority path.

Considering a fixed bandwidth allocation of the priority path, *i.e.* a



(a) $A_{\text{priority}} + A_{\text{mitigation}}$ fixed



(b) A_{priority} fixed

Figure 3.19 – Impact of the mitigation allocation ($BA_{\text{mitigation}}$) on the Legitimate Traffic Reception

constant $BA_{\text{mitigation}}$ (Fig. 3.19b), we notice that the value of *Reception* of legitimate traffic rises with the bandwidth allocated to the mitigation path, *e.g.* a $BA_{\text{mitigation}}$ of 10% only add a few percents to the *Reception*, such that curves representing a $BA_{\text{mitigation}}$ of 0% and 10% are superposed. In fact, as the constraint defined by $BA_{\text{mitigation}}$ in Equation 3.8 is fixed, we generate the same tBF for all experiments. Hence the traffic forwarded through the priority path is the same regardless the $BA_{\text{mitigation}}$ value. In that case increasing the bandwidth allocated to the mitigation path improve the reception of the legitimate traffic.

3.6.5 Efficiency Evaluation

Figure 3.20 show the efficiency of our solution compared to a weighted load-balancing that distributes traffic through legitimate and malicious paths. We obtained the results under the following conditions:

1. the Target Signature Length is null, *i.e.* the mechanism only load-balances traffic according to the source and destination IPs,
2. legitimate traffic volume is equal to 62% of the priority allocation, *i.e.* a common link usage,
3. the allocation of mitigation link is equal to 20% of the bandwidth allocated to both priority and mitigation links,
4. the attack volume varies between one and 50 times the volume of the legitimate traffic volume.

The blue continuous line depicts the application on our mitigation solution under this context. We extracted this curve from Figure 3.19a. The red broken line shows the case where a naive mitigation is applied (*cf.* next paragraph), *i.e.* a basic load-balancing between priority link and mitigation link to which we have allocated 20% of the overall egress bandwidth.

We assume that a network, implementing a naive mitigation, distributes flows through both priority and mitigation paths according to a weight-based and deterministic load-balancing algorithm [108]. A deterministic scheme enforces that the flow is forwarded according to a subset of its fields, for instance source and destination IP addresses. It obviously excludes algorithms such as Round-Robin. Weights assigned to each path in the deterministic load-balancing scheme are inferred from the path bandwidth allocation. Then the traffic is constrained to their allocation with the same throttling algorithm than our solution.

With naive mitigation, the reception of legitimate traffic is equal to 0.85 when the attack volume is equal to the legitimate volume, *i.e.* the bandwidth of incoming legitimate and attack traffic is equal to the overall egress allocation. The naive load-balancing scheme is indeed a flow-based scheme

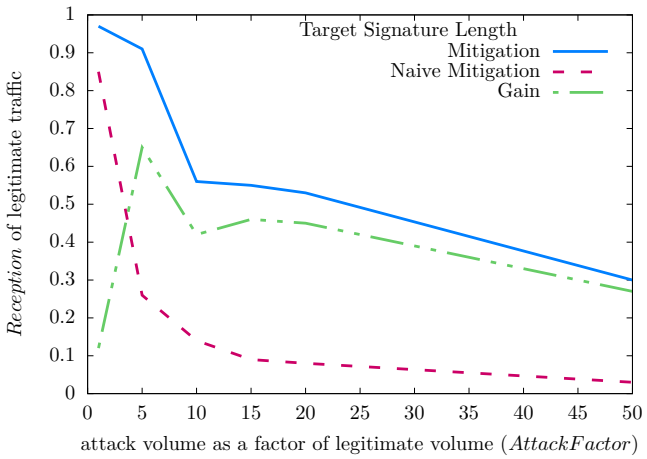


Figure 3.20 – Comparison of the legitimate traffic reception with and without mitigation

(conversely to schemes such as Round-Robin) and the distribution of volume over outgoing paths is highly dependent on the volume of each flow. In that case, the traffic load-balanced through the mitigation link exceed its allocation and is then throttled. The reception of legitimate traffic of the naive use case, however, collapses to 9% for attacks that reach up to 15 times the volume of legitimate traffic. The *Reception* then linearly decreases with the attack volume. The gain of our mitigation mechanism (*i.e.* the difference between both "Mitigation" and "Naive Mitigation" curves) is less valuable for small attacks which volume is equal to the legitimate volume as it only adds 10% of legitimate traffic reception. Nonetheless there is a greater interest to implement the mitigation for more significant attacks. It allows to save at least 40% more of legitimate traffic for attacks with a volume between 5 and 20 times the legitimate volume. The gain slowly decreases to reach 27% of legitimate traffic saved, considering an $AttackFactor$ of 50.

3.7 Discussion

While we introduced the target signature as a pre-filter that matches packet destination IP addresses, to enhance the capacity of the mitigation label in selecting suspicious flows and preserve legitimate flows, results show that it does not improve the mechanism under the chosen context. Increasing the target signature length implies that each tBF counter handle more flows. Indeed, we reduce the hash domain, while the number of flows, either suspicious or legitimate, remains the same. It leads then the filtering to be less precise. Experiments show that the effect of pre-filtering the tBF with a Bloom filter (the Target Signature) does not counteracts the effect of reduc-

ing the hash domain.

Experiments on the mitigation path allocation show it impacts on the reception of legitimate traffic at the egress link. However, concerning a fixed allocation of the overall bandwidth, value of *Reception* decreases as the allocation of mitigation path increases. Conversely, regarding a fixed allocation of the priority link bandwidth, the gain increases with the $BA_{\text{mitigation}}$ value. From a network service provider point of view, this means that in order to improve the legitimate traffic reception, it should reserve a certain amount of bandwidth to allocate mitigation path for a customer. It then cannot be taken as a part of the nominal customer bandwidth. Moreover, the choice of the $BA_{\text{mitigation}}$ may be conditioned by both the additional cost of allocating bandwidth for suspicious traffic and the cost of legitimate traffic loss. The cost of both allocating bandwidth for suspicious traffic by increasing the customer bandwidth and dropping legitimate traffic should be studied in order to choose the $BA_{\text{mitigation}}$.

The comparison between the Mitigation Label-based load-balancing scheme and the naive load-balancing (*i.e.* without mitigation applied) show potential benefit on the reception of legitimate traffic of the tBF. We tend to reduce the impact of the QoS mechanism (*i.e.* the rate-limiting algorithm) by using the same algorithm for both cases, *i.e.* with the Mitigation Label-based and naive mitigation. The solution is particularly efficient for medium strength attacks, *i.e.* which volume is between 5 and 20 times the volume of legitimate traffic.

We should point out that for a stateful protocol such as TCP, the impact of a packet loss may be more costly. It will trigger mechanisms such as retransmission and can impact transmission window size, *etc.* It may then induce delays or even session resets. Such aspects were not taken into account in our experiments. In addition, in our simulated environment, due to the static behavior of capture replays, sources seem to continue to send consecutive TCP packets in the stream despite any packet losses. In a real network, the session may end within after a timeout period, *i.e.* following a number of successive packet drops that prevent the destination from acknowledging.

We compared our solution to a weighted load-balancing scheme, *i.e.* a packet hash is used to map the packet to either mitigation or priority LSPs according to weights associated to LSP bandwidths. Considering a given value of $BA_{\text{mitigation}}$, our solution has better results regardless of the attack factor. For example, with a $BA_{\text{mitigation}}$ of 0%, 10% and 20%, we found that our solution increases the reception of legitimate traffic up to 50% for an attack factor of 20. In fact, as the simple load-balancing scheme does not prioritize legitimate traffic, the value of *Reception* quickly drops with the attack factor.

Our solution looks similar as the IDR [95] as it is based on Bloom Filters. However, the IDR authors aims at detecting the DDoS attack with the help

of multiple Counting Bloom Filters. This means that they want to select attack flows with accuracy and the maximize true positive, *i.e.* attack flows considered as suspicious by Bloom Filters. Our motivation is the opposite as we tend to maximize true negative, *i.e.* legitimate flows not considered as suspicious by the tBF. We provide a quantitative comparison of both results in Table 3.6. It depicts the detection metric, *i.e.* the ratio between the volume of attack detected as suspicious by the IDR (or the tBF) and the incoming attack traffic volume. In our case we consider true positive traffic volume (attack flows that are going to be forwarded through the mitigation path) before being throttled in the same context as the experiments described in Subsection 3.6.3. We extract values for the experiments of IDR that described a network on which an attack hits 3 ingress routers and flows towards a single egress router. Chan *et al.* reported detection values for each intrusion detection router, including edge and core routers. In order to match our architecture, we only consider the detection val- ues on the 3 ingress routers hits by the attack. These values are among the higher. Considering most voluminous attacks, our solution has a higher detection than with the Intrusion Detection Router. In fact, the tBF generation depends on the satisfaction of the priority path bandwidth constraint by the negative traffic, *i.e.* traffic load-balanced through the priority link. However, minimal detection rate of ingress IDRs is better than with our solution considering smaller attacks, *e.g.* which attack volume is equal to the legitimate traffic volume. The multiple Bloom filters the IDR makes use and their larger size (256bits compared to the 20-bit tBF with a null T SL) allow the detection router to select with more accuracy flooding flows.

Table 3.6 – Comparison of true positive probabilities (p_{TP}) with IDR

	IDR	**ML** (tBF only)
min p_{TP}	0.4	0.42 ($AttackFactor = 1$)
max p_{TP}	0.85	0.95 ($AttackFactor = 50$)
average p_{TP}	0.6	0.85

Whitaker *et al.* [153] proposes to carry the representation of the route taken by a packet into a packet. They then add a new protocol layer to achieve the information advertisement. The related protocol header contains the Bloom filter-based route record. By using the MPLS label to carry the flow filter, our solution does not require the use of such a new protocol. Hence routers only have to implement the new mitigation label-based flow to LSP mapping. HADEGA proposes to use SNORT signatures to achieve the flow to LSP mapping. As our proposal only relies on the mitigation label to choose on which path the packet has to be forwarded, this is a MPLS router-based decision.

Chapter 4

Enhancing Blacklist-based Mitigation

Contents

The high volume of recent DDoS attacks forces some network operators to use blacklists (*i.e.* a list of flows to filter, and whose definition is often limited to the OSI Layer 3 and 4). Blacklists defined over the OSI Layer 3 and 4 fields are computationally efficient mechanisms. They filter a large amount of traffic in order to drastically reduce the volume of a volumetric DDoS attack. Furthermore, blacklists can be distributed on a wide range of network equipment.

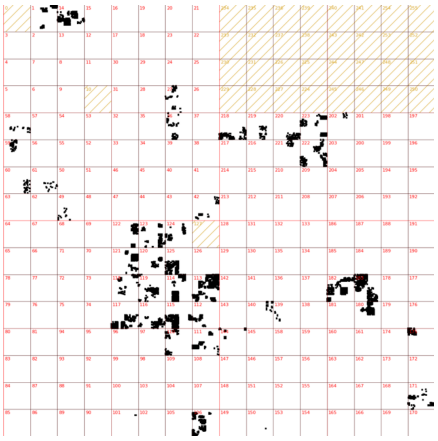
However, equipment on which these blacklists are deployed have limited resources (CPU, memory, ...). We study the performance of filters based on such blacklists within contexts that reflect hardware limitations of network equipment as well as contextual details made available to the administrator. This contextual information impacts the blacklist generation and therefore their efficiency, both in terms of filtered attack traffic and collateral damages. By using these contexts, we intend to study blacklists in an operational point of view, *i.e.* how to filter traffic given information an operator really have access to.

This chapter first introduces features that model the mitigation context (*e.g.* network monitoring, ...) and blacklist configuration parameters (granularity, number of rules, ...). We then describe how we build blacklists from those inputs and define blacklist generation strategies. Subsequently, we assess these strategies within several contexts. We finally make recommendations on the generation and use of blacklists for volumetric DDoS attack mitigation.

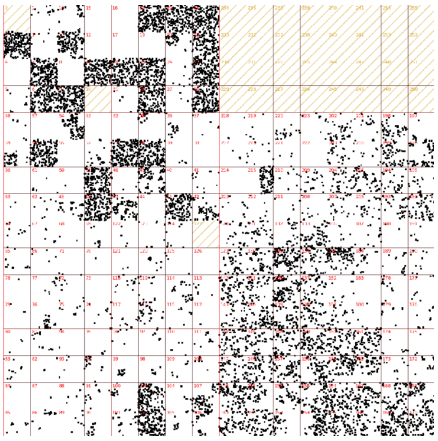
4.1 Introduction to Blacklist: Limitations and Requirements

4.1.1 Threat Landscape

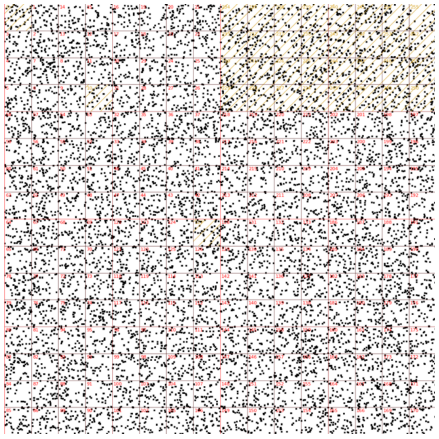
A blacklist is an Access Control List (see Section 2.3.2), where all rules drop traffic. More specifically, all network traffic tested against a blacklist is accepted (*i.e.*, forwarded, routed, *etc.*) except those that match one of its rules. This type of countermeasure is often recommended in case of DDoS attacks [129, 130, 131]. Such recommended blacklist implementation is partially or totally based on the source IP address of network packets. However, blacklist using source IP addresses is mostly inefficient against spoofed IP attacks. An attacker mimics IP addresses of a critical or renowned service, such that blacklisting these spoofed IP addresses will result in dropping all traffic from these services while they are not currently involved in the attack. Figure 4.1a shows the distribution of source IPs during an attack, as reported by Cloudflare [154]. The author was not able to determine whether the attack was spoofed or not, as source IPs were localized, which could also be the case with a botnet-based attack.



(a) Undecidable IP source distribution



(b) Spoofed IP addresses in a limited IP space



(c) Spoofed IP addresses distributed in the whole IP space

Figure 4.1 – Example of dispersion of source IP addresses [154]

On the contrary, random IP spoofing depicts techniques where source IPs are randomly selected from one or several IP ranges. Figure 4.1c and Figure 4.1b show distribution of source IPs during an attack, within the whole IP space and a restricted IP space respectively. Source IPs are much more distributed among the restricted IP space, studying their distribution can be helpful to determine if we faced spoofed direct attacks. This is due to the fact that a malicious source (*e.g.*, bot) can change the source IP. For example, a bot can modify the source IP for each packet.

In that case, the traffic sees an increase of sources with very few packets, *i.e.* the entropy of the source IP header is high. This has a critical implication for static filters (*i.e.* the adaptation to traffic changes requires reconfiguration) such as blacklists: bots are likely switching the spoofed source IP address according to attacker order, so that current filter which contains previous addresses does not filter the new ones. New source IP addresses appears in the attack traffic, that would be also added to the filter. As a result, the mechanism ends up filtering the entire spoofed IP space, causing important collateral damage. Filtering spoofed attack traffic based on the source IPs or a combination of source IPs and other fields is thus inefficient and harmful. Thus, the use of source IP-based blacklists and more generally source IP-based filters is not applicable to spoofed traffic.

Considering volumetric DDoS attacks, we can, in general, mitigate the non spoofed part of an amplification attack (*cf.* Figure 4.2) and stateful floods as exposed in Table 2.1. Remarkably, some direct UDP-floods do not use spoofing, so that they can be filtered using blacklist. For example, in 2016, the Mirai botnet [155], comprised of IoT bots, hit the OVH company [1]. The accumulated volume of malicious traffic at OVH network's entry points reached around 1Tbps. More than 145,000 simultaneous non spoofed sources, particularly IoT devices, were identified as participants of this attack.

4.1.2 Typical Operating Environment

Although attack detection is out of the scope of this thesis, the identification of attack traffic is intrinsically tied to the detection mechanism capabilities. Figure 4.3 shows a generic security architecture, not restricted to the network security. Probes first send events, for example logs or IDS alerts to an Event Manager (EM). Reported events are not restricted to malicious incidents, and benign events can be sent as well. The EM can then apply an intelligent treatment, *e.g.* correlate events, so that it enriches events and produces alerts. The Policy Decision Point (PDP) takes as input alerts, computes possible reactions and selects the one or more candidates. Mitigation orders are then pushed to the Policy Enforcement Point (PEP), such as firewalls or routers. This generic workflow encompasses the simplest scenario, where a web admin detects that its website is down due to a flood of requests and

calls its hosting provider to request help.

Selection of mitigation orders depends on at least two parameters, the alert (or set of alerts), that refer to the attack requiring mitigation (*cf.* Subsection 4.1.3), and the PEP capabilities (*cf.* Subsection 4.1.4).

4.1.3 Requirements: Identification of Attack

Blacklist usage, not restricted to the network area, is fundamentally based upon the identification of items to block, *i.e.* the items to add to the list of components to filter. In the area of networking, an item refers to network traffic, which can be identified with Layer 3 and 4 header fields such as IP addresses, the protocol and ports. Thus, the alert pushed to the PDP in Figure 4.3 has to contain the identification of the traffic to filter.

Below we examine three common alert formats (EVE, IDMEF, DOTS) and show that for all three these minimum requirements are part of the base definition of an alert.

Extensible Event (EVE)[1] is a data exchange format provided by Suricata[2]. Suricata is an IDS that is able to respond to threats, *i.e.*, it is also an IPS. It inspects network packets and analyses them using pattern-based rules. Alerts produced for traffic matching a rule can be expressed in various formats, such as syslog [156], or json [3] using the EVE format. Listing 4.1 shows an example of an alert concerning a potential DNS amplification attempt from a local network targeting an external IP. At least, an EVE-formatted event identifies the source(s) and target of a threat as well as the Layer 4 protocol. It may also specify the source and destination Layer 4 ports if the protocol defines such headers fields, such as UDP used in amplification DDoS attacks.

Listing 4.1 – Example of Extensible Event (EVE formatted) Alert produced by Suricata IDS facing an Amplification DDoS

```
{
  "timestamp": "2009-11-24T21:27:09.534255",
  "event_type": "alert",
  "src_ip": "192.168.2.7",
  "src_port": 53,
  "dest_ip": "x.x.250.50",
  "dest_port": 80,
  "proto": "TCP",
```

[1] EVE v4.0.1, http://suricata.readthedocs.io/en/latest/output/eve/index.html

[2] Suricata, https://suricata-ids.org/

[3] json format, https://json.org/

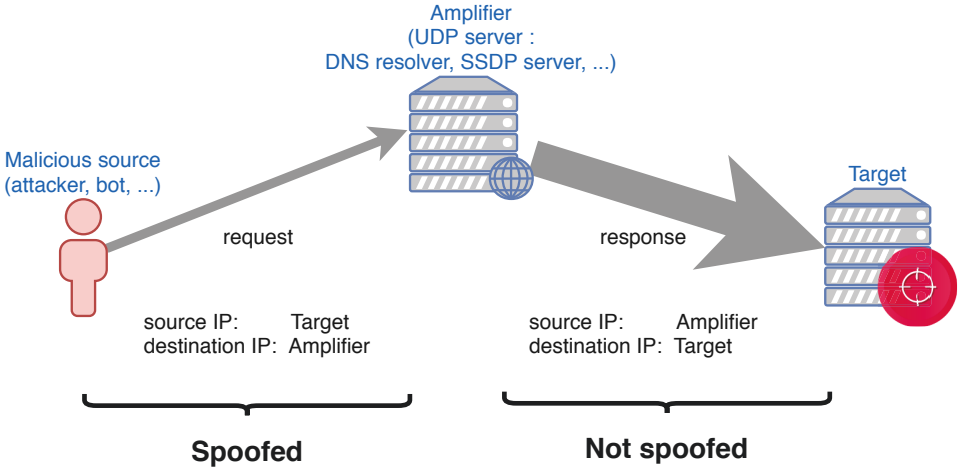


Figure 4.2 – Amplification attack traffic

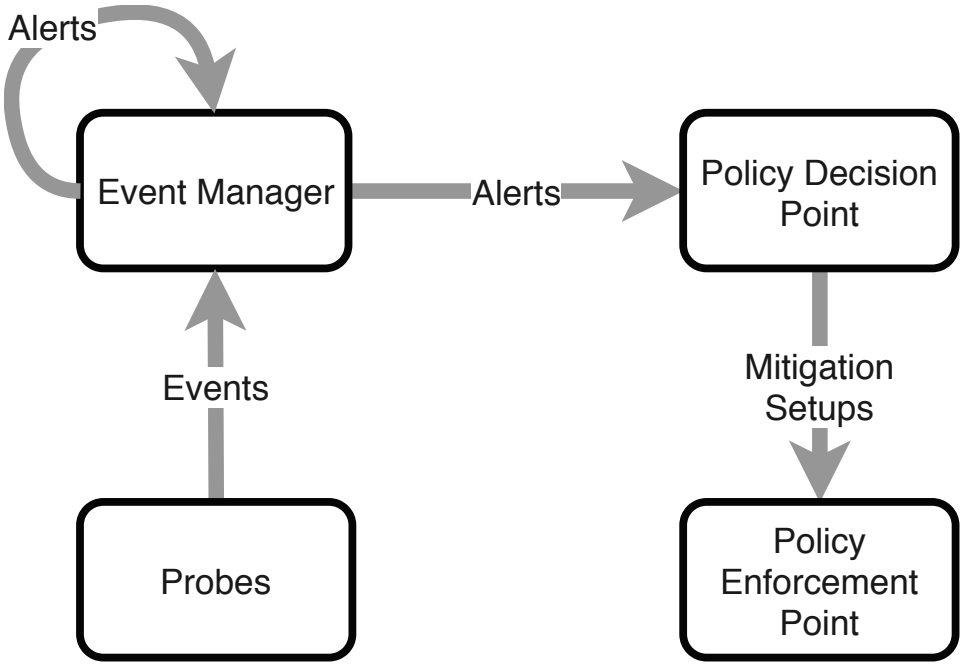


Figure 4.3 – Generic security architecture

```
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id" :2016017,
    "rev": 6,
    "signature": "ET CURRENT_EVENTS DNS Amplification
        Attack Outbound",
    "category": "Potentially Bad Traffic",
    "severity": 2
  }
}
```

Intrusion Detection Message Exchange Format (IDMEF) [144] is an XML-based data format, that can describe both alert and heartbeat messages. A wide range of security systems have the ability to export IDMEF formatted events, such as Network IDSs (*e.g.*, Snort[4]), Host IDSs (*e.g.*, OSSEC[5]) or SIEMs (*e.g.*, Prelude[6]). By default, IDMEF does not concretely specify the source and target of the event but the service (*cf.* Table 4.1), the node, the user or the process. The node carries the IP address while the service carries the port.

Distributed Denial of Service Open Threat Signaling (DOTS)[7] is an active and still in progress standard for real time DDoS related information exchange. It includes the entire life cycle of a threat, from the request for mitigation to reaction management and to threat details update. It specifies two communication channels, namely signal and data channels. The signal channel [157] is intended to convey mandatory information (*e.g.*, heartbeats, mitigation requests and status). The data channel [158] aims at transmitting details, such as mitigation configuration through ACL rules. While the signal channel must be resilient to the worst case, such as volumetric DDoS attacks, the data channel does not require to achieve data exchange under attack condition. In that respect, the signal channel allows only to advertise the target, and does not provide means to specify attack sources. Target can be identified by a combination of a prefix, Layer 4 protocol and ports, and a domain name. The data channel can convey attack sources using almost all IP Layer fields and fields of Layer 4 protocols (UDP, TCP, ICMP). The traffic to be filtered is assumed to be flows from these defined sources and towards the target advertised through the signal channel.

[4] https://www.snort.org/
[5] http://www.ossec.net/
[6] http://www.prelude-siem.com/
[7] https://datatracker.ietf.org/wg/dots/about/

	EVE	IDMEF	DOTS
source identification	• IP address • *Layer 4 port*	• one or more node (name, address) • one or more user • one or more process • one or more service (name, protocol, port, ...) • *whether spoofed or not*	• *Layer 3 headers* • *UDP, TCP, ICMP header fields*
target identification	• IP address • *Layer 4 port*	• zero or more node (name, address) • zero or more user • zero or more process • zero or more service (name, protocol, port, ...) • *whether spoofed or not*	• *IP address* • *Layer 3 protocol* • *Layer 4 ports* • *domain name*
Impact assessment	signature dependent severity index	• impact rating • confidence index	
Additional traffic details	Layer 4 protocol	any other information that probe can provide	

parameter: optional parameter

Table 4.1 – Threat identification in some event signaling formats

	Filtered Traffic	Forwarded Traffic
Attack Traffic	True Positive	False Negative
Legitimate Traffic	False Positive	True Negative

Table 4.2 – Filtering traffic terminology

4.1.4 Characterization of Attack Traffic: Granularity of Mitigation

The definition of ACL rules is in the process of being standardized [159] using the Yang data model [160]. Thus, a traffic filter rule may include header fields among the following:

- Ethernet MAC addresses
- IP (version 4 and 6) header fields such as DSCP, ECN, length, TTL, Layer 4 protocol, source and destination;
- various UDP, TCP and ICMP header fields, for example TCP and UDP ports.

However not all equipment that implement ACLs are able to filter with such a fine granularity. Moreover, in order to distribute defenses upstream from the choke point caused by a volumetric DDoS attack, the attack traffic should be pruned as soon as possible, *i.e.*, close to the network entry, to avoid the funneling effect. This should be done using all equipment available, including equipment that are not primarily intended to achieve traffic filtering, such as routers. Indeed, as detailed in the state of the art (*cf.* Chapter 2), the use, or implementation of new filtering mechanisms in router has been widely studied. Current off-the-shelf routers have various granularity, but they can all filter at least based on source and destination IPs as well as the Layer 4 protocol and the UDP, TCP source and destination ports.

Table 4.2 shows commonly used terminology for traffic filtering. True positives refer to attack traffic being truly filtered by the ACL. Conversely false positive denotes legitimate traffic being unfairly dropped by the filter, *i.e.*, collateral damage. It may be caused by a too coarse filter or legitimate traffic that overlaps with attack traffic, *etc.* False and true negative, respectively refer to attack and legitimate traffic that does not match the filter and is then forwarded.

4.2 Blacklist Generation Problem Characterization

Broadly speaking, response to an attack and selection of the mitigation may be difficult. To be as efficient as possible, mitigation configuration requires to process heterogeneous information. Efficiency is a subjective concept, which

can be measured in different ways, as stated in Section 2.2. However, considering DDoS attacks, an efficient mitigation mechanism has to maintain the availability of the service to legitimate users with an acceptable user experience. Regardless of the user experience, the mitigation efficiency may be summed up as follows: guaranteeing the availability of the service by maximizing the attack traffic filtering, so that link saturation is avoided, while preventing collateral damage to customers. This simple efficiency definition ignores that mitigation itself can cause an extra burden, such as inducing additional delay to traffic, which may also degrade the perception of the user experience.

In this chapter, we consider blacklists as a response to volumetric DDoS attacks. We can identify three distinct objectives for such a response:

- one wants to filter as much attack traffic as possible. For example, router's blacklists or a firewall are one's single line of defense.
- one wants to filter just enough attack traffic to prevent link saturation, letting the remaining traffic to flow towards the target. That may be the policy for a transit or hosting provider who only intends to protect their own infrastructure.
- one wants to filter enough traffic to prevent link saturation, while the remaining traffic is redirected to a mitigation mechanism that can clean traffic with a finer granularity than ACLs, as stated by Pack *et al.* [39]. An ISP who sells traffic protection may implement such a policy.

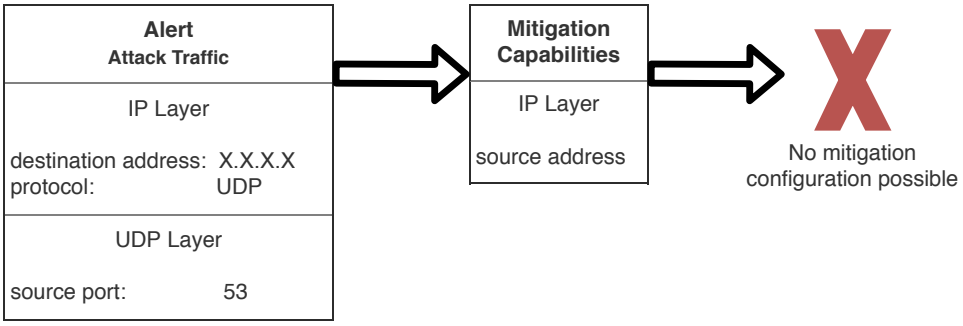
As a consequence, maximizing attack traffic being dropped means that at least enough traffic (or subset of traffic) is filtered to avoid funneling effect and prevent link saturation.

4.2.1 Maximizing Filtered Attack Traffic

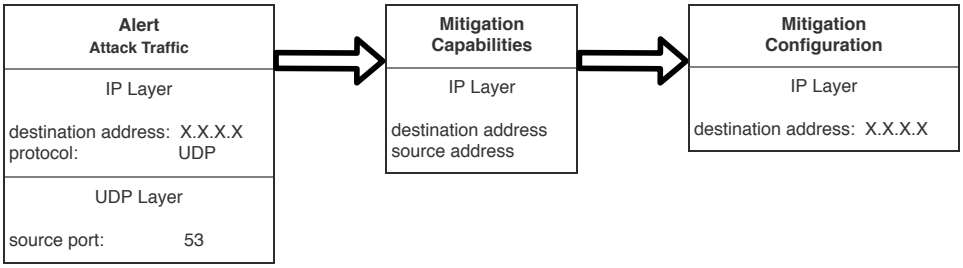
In volumetric DDoS attacks, traffic filtering should at least prevent malicious traffic from creating the denial of service condition, *i.e.*, the network congestion through link saturation. Therefore, we should maximize the amount of malicious traffic being filtered, so that the congestion does not occur. Especially, we should maximize the volume of malicious traffic being dropped.

Configuring filtering based on malicious traffic characteristics contained in alerts may be difficult. In fact, the granularity of the identified attack traffic may be different from the granularity of the filter. Figure 4.4 depicts such an issue. An alert contains a notification about a DNS amplification attack and coarsely identifies the attack traffic as UDP traffic destined to the target IP X.X.X.X from source port 53. If the mitigation granularity mismatch the

attack traffic characteristics, as depicted in Figure 4.4a, where the equipment only allows source-based ACLs, it is almost impossible to configure the mitigation. Otherwise, the mitigation is configured with the intersection of the alert granularity and the mitigation equipment capabilities as shown is Figure 4.4b.



(a) Mismatch between attack characterization and mitigation capabilities



(b) Partial overlap between attack characterization and mitigation capabilities

Figure 4.4 – Examples of mitigation configuration

As a consequence, ACL definition should match the alert definition, where ACL is defined using common characteristics of the alert and mitigation capabilities. Given that the alert characterizes malicious traffic using i packet header fields, namely F_i (*e.g.*, IP addresses, Layer 4 protocol, ports, ...), the mitigation only allows configuring ACL using F_j, a subset of the packet header fields F_i, mitigation can then at most be applied based on the subset of packet fields F_j. Discarding a field in the mitigation configuration leads to filtering packets whatever the value of the fields $F_i \setminus F_j$, *i.e.* the corresponding filters would use wildcards for the missing fields.

Operational constraints, such as the number of rules in an ACL, *cf.*, Section 2.3.2, may also lead in reducing the filter granularity. For example, the state of the art, promotes the use of source IP aggregation to reduce the number of rules [42, 36, 39, 43].

4.2.2 Minimizing Collateral Damages

The drawback of maximizing the amount of malicious traffic that is being filtered is that it also increases the amount of legitimate traffic that will be dropped by the mitigation. Indeed, collateral damages are induced by the overlap between the legitimate traffic and the definition of mitigation rules. Using wildcards for an operational reason (*e.g.*, reduction of the number of rules, granularity mismatch between alert and mitigation) potentially induces more legitimate traffic to be matched by the filter.

As stated in the literature, we need information about legitimate traffic to reduce the collateral damage induced by the granularity of the mitigation. The information (*e.g.* traffic telemetry), however, does not differentiate legitimate and attack traffic.

We highlight the fact that information used to maximize filtered attack traffic and minimize collateral damages are obtained from different sources. In fact, we use alerts to obtain information about malicious traffic and monitoring system information such as flow records to find information about legitimate traffic. Two sources imply the possibility of different granularities of information, such that the alert granularity may mismatch the granularity of flow monitoring records. Especially given that monitoring flow records may use traffic aggregation [161], for example, aggregation of source or destination IP prefixes or both. We must be able to handle the differing granularities when processing both information types to generate the ACLs.

4.2.3 Characterization of Network Visibility

While literature has proposed algorithms to generate blacklist filters with a realistic number of rules, these algorithms proposed in literature, however, are not applicable in all networks due to the requirements in terms of information availability. Furthermore, they do not evaluate the efficiency of their approach in regard of the amount of information available concerning the protected network. For example, Soldo *et al.* in [36] assume that sources of legitimate traffic are known or at least that the granularity of information about both the overall and malicious traffic are the same, which is not always the case. Especially in the case of ISPs who have no knowledge about the services their customers provide or who use traffic aggregation in their monitoring system.

The visibility an operator has on his network (*e.g.*, amount of information, level of details) is highly dependent on the monitoring policy, equipment, *etc.* Similarly, attack details available in the alert depend on the equipment that detects the threat, *e.g.* by analyzing the monitoring information or using dedicated equipment. In addition, the dedicated detection mechanisms do not provide an equal level of details. We therefore define three scenarios describing different network visibility levels and summarize them

in Table 4.3.

Table 4.3 – Generic information availability-driven scenarios

Scenario	Description	Malicious traffic identification	Malicious traffic telemetries	Monitoring telemetries
1	Minimum requirement	Yes	No	No
2	Enhanced detection	Yes	Yes	No
3	Full network visibility	Yes	Yes	Yes

The *minimum requirement* scenario is a situation where only the minimum information required to generate a blacklist, *i.e.* the characterization of malicious traffic, is available an is extracted from the alert (*cf.* Subsection 4.1.3).

The *enhanced detection* scenario describes the context where an operator has access to more detailed information about malicious traffic than solely the identification. He may then be able to retrieve telemetries for malicious traffic, for example from the detection mechanism. These features should be collected at the same granularity as the detection. For example, they may also be included in the alert.

Our *full network visibility* scenario refers to the use of monitoring information to reduce the amount of collateral damages. Monitoring information are related to the entire network traffic and, as such it does not differentiate legitimate from malicious aggregates. Off-the-shelf mechanisms, such as NetFlow [146], sFlow [147] or IPFIX [148], are able to provide metrics such as volumetries for traffic aggregates.

However, because the granularity of the definition of the malicious traffic is defined by the detection system, and since the monitoring record (including telemetries) granularity depends on the monitoring system configuration, the granularities of the detection system and the monitoring system may not coincide. For example, an alert can contain a list of the following 5-uplet: source and destination IPs, IP protocol and source and destination ports, while the monitoring system collects traffic features only for destination IPs. We will study the impact of information availability on the blacklist efficiency with regard to these scenarios.

4.2.4 Network Visibility Cost

The underlying goal of characterizing the network visibility required by a mitigation mechanism is to quantify its cost. The visibility cost may then be

used later to study the balance between the cost induced by legitimate traffic losses and the cost of mitigation. Indeed, a better visibility on the network may allow achieve a finer mitigation and then reduce collateral damages. However, we argue that from an operational point of view, such a solution is not suitable, when maintaining such a level of visibility on the network costs more than the potential legitimate traffic losses.

We particularly focus on the cost of network monitoring. "Cost" does not only refer to the monetary price but can also refer to any metrics that are affected by maintaining the network visibility expressed in Equation (4.1).

$$cost = cost_{flow\ collection} + cost_{record\ transfer} + cost_{record\ storage} \tag{4.1}$$

We identify three monitoring actions that may impact network operational cost, namely:

- the collection of network flow records ($cost_{flow\ collection}$). An equipment that collects flow telemetries has to keep track of flows and counters. While this consumed CPU cycles and memory [162] on older equipment, new ones implement flow collection in hardware such that the induced CPU load overhead is minimized [163].
- the transfer of flow records to the collector that centralizes records from many devices ($cost_{record\ transfer}$). The bandwidth consumed depends on several parameters such as the number of flows seen by an equipment, the granularity of a flow, the level of details of statistics.
- the storage of flow records ($cost_{record\ storage}$). A collector stores flow statistic for a given period of time so that it is able to provide a network usage history.

Online tools are available to estimate the bandwidth consumption and required storage [8].

4.3 Application: Blacklist Generation Approaches to IP level

In the remainder of this chapter, we place ourselves in the context of a network operator, such as an ISP, transit provider, or simply a hosting provider. As a consequence, we consider the IP Layer granularity as a trade-off between the operational constraints (topology, cost, *etc.*) and mitigation capabilities. We study blacklist efficiency in an operational context, without

[8] Netflow Scrutinizer bandwidth and storage estimation tool https://www.plixer.com/resources/netflow-bandwidth-calculator/

adding requirements or constraints to the existing security and networking environment. The operational environment includes a security monitoring architecture that produces alerts (*cf.* Subsection 4.1.1), an equipment that implement blacklists and a network traffic monitoring system which provides network traffic telemetries (*cf.* Subsection 4.2.3).

Thus, Subsection 4.3.1 first defines the terminology used for the refinement of our application context. Subsection 4.3.2 states the assumptions induced by the limitation to IP Layer and completes the definition of generic scenarios outlined in Subsection 4.2.3.

4.3.1 Definitions

We provide below definition of some concepts used in the remainder of this chapter.

Detection system is any system capable of detecting volumetric DDoS attacks and reporting the source and destination addresses participating in the attack. A detection system can be external to the network being monitored.

Monitoring system is any system providing network telemetries for the monitored system. For our needs, we expect the telemetry to include at least traffic volumes between source and destination addresses, eventually with some level of aggregation.

Aggregate is a network address prefix aggregate as used in route aggregation.

IPf is a stream of packets, sharing the tuple < source IP, destination IP >. Defined this way, the flow includes only one direction of traffic and for example a TCP connection will result in two IP flows.

MALagg is an aggregate of traffic, defined as a set of one or more IPf that, according to a detection mechanism, contains malicious traffic. It should be noted that due to the coarse granularity of its definition, such an IPf can also contain legitimate traffic.

MONagg is a set of one or more IPf observed by a monitoring system. Depending on the monitoring system's configuration, it reports MONaggs with a particular granularity, eventually aggregating source and/or destination addresses. In other words, MONagg are defined by the source and destination network addresses (both using CIDR), where the source and destination netmasks are fixed.

4.3.2 Assumptions and Induced Scenarios

Alert Considering the threat we focus on, *i.e.* volumetric DDoS attacks, we can then reasonably assume that we are able to retrieve at least the target IP prefixes, that obviously include single IP addresses. If a domain name is provided instead, the resolved IP would be taken as the target.

We also expect to retrieve an exhaustive list of source IP addresses, or at least, a list of IP addresses that are the sources of a large part of the attack volume. Indeed, it may be difficult for a detection mechanism to have a detection rate (*i.e.* recall as expressed in Equation (4.2)) of 100%. The impact of the recall is discussed in Subsection 4.5.3.

$$recall = \frac{TP}{TP + FN} \tag{4.2}$$

In the remainder of this chapter, we assume that alerts contain malicious IPf, where destination IPs match attack targets and sources reflect malicious IPs that send traffic to at least one target.

Mitigation We assume that an equipment which implements blacklists are able to filter at least the source and destination IP prefixes. Each filtered aggregate, defined by a source and destination prefix, can then be configured into this equipment. Even routers benefit from this coarse granularity [35]. This allows us to be equipment agnostic.

We do not make any assumption on how ACLs are implemented into the equipment. However, both hardware or software-based implementation have limited number of rules, although the limit in the number of rules of software-based implementation is generally much higher. We denote the maximal number of rules in a blacklist with N.

Induced Scenarios We refine the three information availability-driven scenario defined in Subsection 4.2.3 with regard to the study assumptions.

The malicious traffic is identified by a list of MALagg extracted from alerts and provide by the detection mechanism. In practice, alerts may contain a list of IPf instead, which are defined finer than MALagg. Alerts may also contain telemetries for malicious traffic, which case is depicted in the second and third scenarios. These metrics are collected at the same granularity as the detection, *i.e.* the tuple < source IP, destination IP >. The third scenario also includes the collection of network monitoring records. Monitoring information extracted from the records are provided for MONaggs.

4.3.3 Fundamentals of Blacklists

This section presents a blacklist generation scheme that deals with the problems raised in Subsection 4.1.1 - Threat Landscape, and the context exposed

in Subsection 4.2.3 - Characterization of Network Visibility. As the number of rules in a blacklist is limited, the capability of the ACL to filter malicious traffic depends on the ability of the generation process to aggregate malicious flows, so that the amount of filtered attack traffic is maximized. A workaround to the limitation of the number of blacklist rules would be to increase the amount of traffic to be filtered among the whole traffic, *e.g.*, by shortening the length of the rules source prefix. However, this also probably induces more collateral damage. Consequently, the ACL generation should also tend to minimize the false positives, *i.e.*, legitimate traffic that is being filtered by the ACL.

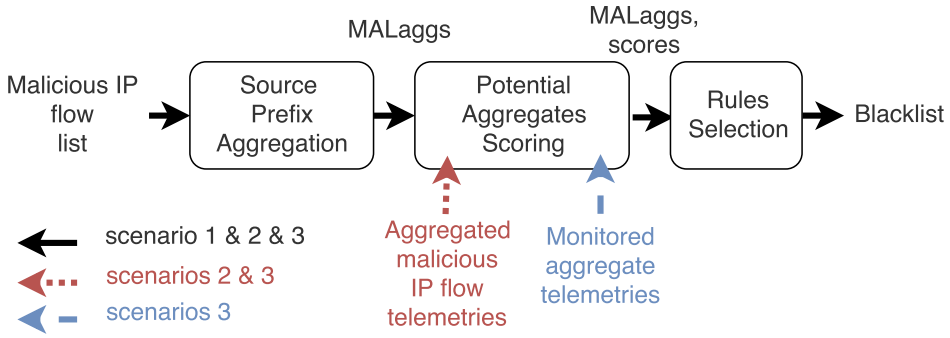


Figure 4.5 – Workflow of ACL generation

The ACL generation process is depicted in Figure 4.5. First, we compute all possible malicious aggregates (MALaggs) for IP flows (IPf) included in the alert (*cf.* Section 4.3.3). Traffic to each destination IP address is treated separately, so that regardless of the filtering granularity (*i.e.*, either based on source IP or both source and destination IPs), only the source IP of malicious flows is aggregated. Second, a score is computed for each of the MALagg (*cf.* Section 4.3.3) by using aggregated malicious IPf telemetries if they are available (scenarios 2 and 3) and monitored aggregates (MONagg) telemetries (scenario 3). Third, the top N MALaggs are selected as rules to form the blacklist (*cf.* Section 4.3.3). Then, the scores are regularly recomputed to maintain up-to-date blacklists, for example every time an alert is received from the detection system.

Source Prefix Aggregation

As widely approved by the literature ([45, 42, 164, 165, 166]), we reduce the number of ACL rules using aggregation of source IP addresses for a given destination IP. We thus maintain a separate list of sources for each target.

We define the *aggregation limit* (AL) as the minimal source prefix length of potential aggregates, so that aggregates have a source prefix length between the AL and 32, *i.e.* a single IP address. Figure 4.6 shows an example of

computed source aggregates for a given destination IP. Considering an *AL* of 26, an alert that contains the following source IPs [1.66.180.12, 1.66.180.13, 1.66.180.50, 1.66.180.60, 1.66.180.201] for a single target results in the following list of possible source aggregates [1.66.180.12/32, 1.66.180.13/32, 1.66.180.50/32, 1.66.180.60/32, 1.66.180.201/32, 1.66.180.12/31, 1.66.180.48/28, 1.66.180.0/26], shown in green in Figure 4.6. The aggregate 1.66.180.0/24 is not included in the MALaggs as the netmask length exceeds the *AL*.

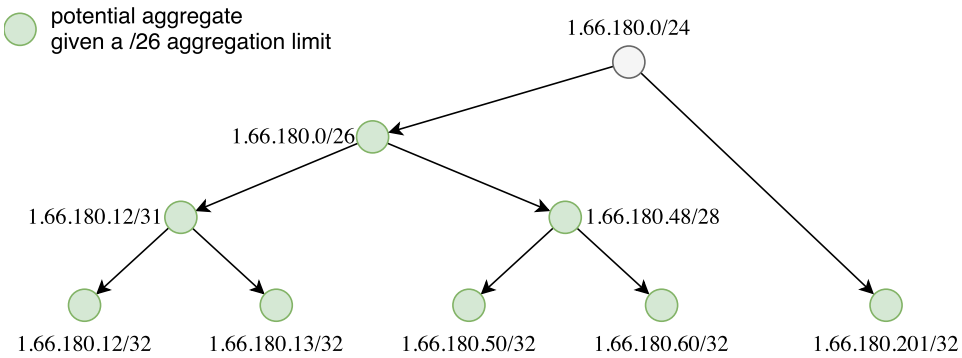


Figure 4.6 – Example of source aggregation tree for a given destination

Malicious Aggregates Scoring

We define three main strategies for scoring MALaggs that aim at dealing with scenarios that only include information about malicious traffic (*cf.* scenarios 1 and 2, Table 4.3). A fourth strategy concerns the last scenario that includes monitoring telemetries. For all strategies, aggregates with high scores are more likely to be added to the ACL. We do not claim that these simple strategies are better than the state of the art. They aim at reflecting how network information can be used and how level of information impacts the filtering.

Scenario 1.a aggregate scoring is relevant to scenario 1 where network operators can only retrieve a list of MALaggs. The generation scheme scores possible aggregates by only taking into account the length of the aggregate's source IP prefix p, as shown in Equation (4.3). Aggregates with a shorter source IP prefix length get a higher score and are more likely to be inserted in the ACL, such that scores are narrowed between 0 and 32.

$$score_{1.a}(p) = 32 - length(p) \tag{4.3}$$

Scenario 1.b aggregate scoring also focuses on scenario 1 where operators only get a list of malicious contributors. The score is equal to the ratio between the number of malicious sources ($\mathcal{MS}$) within an aggregate and the complement to 32 of the aggregate's source netmask length, to which

has been added 1 so that /32 prefixes do not result in a division by 0. In that case, potential aggregates, which include a larger number of malicious sources and/or whose source prefix is small, get a higher score to be ranked higher in the blacklist, in order to minimize collateral damages. As a result, the score, which ranges from 0 to $2^{32} - 1$, is expressed in Equation (4.4).

$$score_{1.b}(p) = \frac{|\mathcal{MS} \cap p|}{(32 - length(p)) + 1} \tag{4.4}$$

Scenario 2 aggregate scoring also takes the malicious IPf telemetries as input. We use the volume of flow as the metric to assess the impact on the bandwidth saturation. The aggregate score, expressed in bytes in Equation (4.5), refers to the volume of malicious traffic towards the target, which source IPs ($\mathcal{MS}$) are included in the aggregate source prefix p. The $malVol(ip)$ function depicts the byte sum of malicious traffic from the source IP (ip) towards the target reported during the last period. As such, the score ranges from 0 to the total volume of attack traffic.

$$score_2(p) = \sum_{ip \in \mathcal{MS} \cap p} malVol(ip) \tag{4.5}$$

Scenario 3 aggregate scoring denotes the use of MONagg telemetries. MALagg's scores, also expressed in bytes and ranged from 0 to the sum of volumes of all malicious IP flows, are obtained by multiplying the score obtained in scenario 2 and the ratio between the volume of malicious traffic and an estimation of the overall traffic within the aggregate p ($overallVol(p)$), as expressed in Equation (4.6). The score increases with the volume of malicious traffic. It also decreases when the volume of legitimate traffic increases, *i.e.* the estimation of the overall volume increases while the volume of malicious traffic remains the same.

$$score_3(p) = score_2(p) \times \frac{score_2(p)}{overallVol(p)} \tag{4.6}$$

The source prefix length of potential rules and monitored aggregates may differ. Hence, $overallVol(p)$ is an estimation. In fact, the length of source prefix (p) of the potential aggregates to filter varies between the aggregation limit and 32, while the source prefix (p_m) length of MONagg is fixed by the monitoring system configuration (*cf.* Subsection 4.2.3). The estimation depends on the value of the source prefixes' length as can be seen in Equation (4.7).

$$overallVol = \begin{cases} \sum_{p_m \subset p} monVol(p_m) & \text{for } length(p) < length(p_m) \\ monVol(p) & \text{for } length(p) = length(p_m) \\ (monVol(p_m) - malVol(p)) & \\ \times \frac{nbHosts(p)}{nbHosts(p_m)} + malVol(p) & \text{otherwise} \end{cases} \tag{4.7}$$

If the prefix of a MONagg is larger than the prefix to filter, the estimation of the overall traffic within p is equal to the sum of the volume of all monitoring aggregates ($monVol$) included in the source prefix to filter p. The estimation is equal to the volume of the monitoring aggregate when the length of the aggregate to filter is equal to the configured prefix length of monitoring aggregates. Otherwise, we estimate the volume of legitimate traffic within the source prefix p towards a given destination. We first assume that remaining traffic volume (*i.e.* $monVol(p_m) - malVol(p)$) is evenly distributed on the highest number of hosts in a subnet of size $length(p_m)$ expressed in Equation (4.8). Then, we add the volume of these sources included in the prefix p to the volume of malicious traffic for this aggregate.

$$nbHosts(p_m) = 2^{32-length(p_m)} \tag{4.8}$$

Rules Selection

Finally, we select the top N rules among all scored potential aggregates to form the blacklist. The process is depicted in Figure 4.7. The potential aggregates are sorted according to their scores in descending order. In the example, scores between parentheses have been set arbitrarily. However, it is possible that the aggregate 1.66.180.12/32 has a score higher than one of its parent aggregates, *e.g.*, 1.66.180.12/30. A legitimate client (*e.g.*, 1.66.180.14) with a large volumetry may reduce the aggregate score of the prefix 1.66.180.12/30. Then, the top N ($N = 3$ in Figure 4.7) are used to generate the ACL. Considering the aggregation mechanism, an overlap is possible only if an aggregate is included in another. Consequently, to avoid wasting rules, an ACL has to be exclusive. Then, when we try to insert in the top N an aggregate that includes or is included in an already inserted aggregate, we keep the aggregate with the smallest prefix length and remove the other. In the example, 1.66.180.50/32 and 1.66.180.48/28 are both in the top 3 aggregates. However, as the second aggregate includes the first one, only 1.66.180.48/28 is kept in the final blacklist.

4.4 Blacklists Efficiency Evaluation

The purpose of this section is

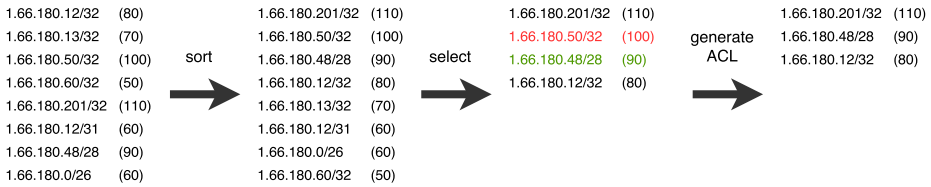


Figure 4.7 – Selection of top 3 rules among potential source aggregates for a given destination

1. to validate the use of monitoring information in the blacklist generation to reduce collateral damages, with regard to the three scenarios we defined in Subsection 4.2.3 and refined in Subsection 4.3.2, and

2. to assess the impact of monitoring information granularity on the mitigation efficiency, *i.e.* on the ability of blacklists to successfully filter attack traffic while allowing legitimate traffic to flow toward its destination.

We conduct simulations in which we generate blacklists using scenario-related strategies and apply resulting filters on traffic captures. We then compare results for each scenario-driven scoring functions. It allows us to study the impact of the levels of available information on the filtering efficiency, *i.e.*, the ability to drop malicious traffic while preserving legitimate flows.

4.4.1 Metrics and Variables

We rely on two commonly used metrics when dealing with filtering, the *true positive rate* (TP_r, also known as sensitivity or recall) and the *false positive rate* (FP_r) in order to assess the scoring strategies. The TP_r evaluates the proportion of malicious traffic that is being filtered by the mechanism, how the filter is able to correctly drop malicious traffic. The generation strategies have been designed to maximize this percentage. Conversely, the FP_r measures the proportion of collateral damages. This allows validating the use of monitoring information to reduce the collateral damages. Both metrics are then well fitted to assess the twofold definition of the efficiency.

The TP_r is obtained as the ratio between the number of filtered malicious IPf and the total amount of malicious flows. Correspondingly, the FP_r is the ratio between the number of filtered legitimate flows and the sum of legitimate flows.

The blacklist construction is tuned using two parameters, the maximal number of rules in a filter (N) and the aggregation limit (AL). Soldo *et al.* [36] used from few hundreds to few thousand rules, while Pack *et al.* [39] solely used up to 50 rules. We should take into account that the more we

insert rules into a router, the longer the mitigation configuration will last. We scripted the automatic insertion of ACL rules into a Cisco 7200 router through the ssh command line interface. Figure 4.8 shows the time to insert rule set as a function of the blacklist size. The loading time depends on the rules already configured in the router, such that the insertion time is not a linear function of the number of rules. However, we have been able to observe pulsing amplification attacks at a French ISP and at a French data center entry. Among the data from last few years, a common attack traffic dynamic shows very short hit periods (around 5 seconds), *i.e.*, traffic peaks, followed by a longer run period without malicious traffic (from 30 to 60 seconds). This attack behavior is used by advanced attackers that aim at bypassing mitigation. As a consequence, the configuration process should not last more than few seconds. We then execute experiments with different values of N between 10 and 500 maximum filtered aggregates in the filter, which insertion lasts up to 4 seconds.

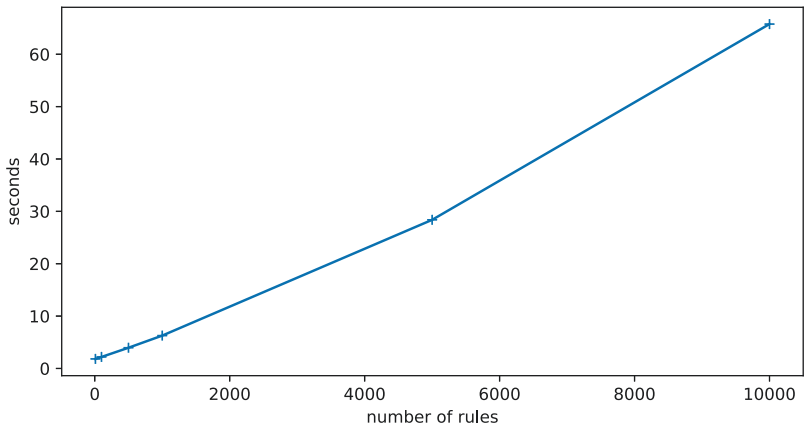


Figure 4.8 – ACL loading time as the function of the number of inserted rules on a router with no rules loaded

Aggregation limit is fixed to either /24 or /8 to study its impact with a short and a long filtered aggregates prefix length. An *AL* of /0 and /32 have not been considered here, as a filtered aggregate/0 will result in dropping the whole traffic. Conversely, filtering with the whole IP (/32 prefix) instead of an aggregate, with at most 500 rules is likely to block very few malicious IP flows. For example, considering an attack which consists of 200,000 malicious IP flows, at most 0.25% of malicious IPf to be filtered which can be considered as pointless. The usability of filtering /32 prefixes depends on the aggressivity of the top malicious contributors. For example, if 200 IP flows are responsible for 80% of the malicious traffic volume, filtering IP addresses (/32 prefixes) is relevant. However, for reasons of clarity, we decided not to

include IP addresses.

We consider two configurations for the MONagg's granularity reported by the monitoring system. The first one defines records for destination IPs, *i.e.* traffic metrics are reported for < `/0 source prefix, /32 destination prefix` > aggregates. This kind of monitoring configuration may be used for networks where each customer is identified by the destination IP such as data centers. The second finer granularity is defined by the tuple < `/24 source prefix, /32 destination prefix` >. That can be used, for example, by ISPs, so that records match the largest common inter-AS BGP prefixes advertisements [167].

4.4.2 Constant Bitrate Attack Traffic Model

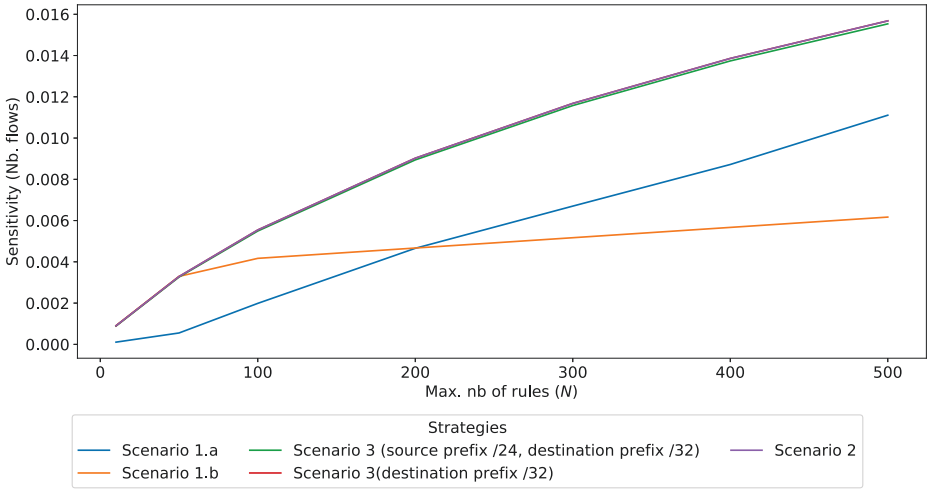
Traffic Model

We use real legitimate traffic from the MAWI data set [152] as legitimate traffic superimposed with generated attack traffic. Traffic has been captured on February 2017 during 15 minutes on a transit link and has been cleaned from attack traffic [9]. We generate 10 different attack traffics, such that each generated attack trace is superimposed to the legitimate capture.

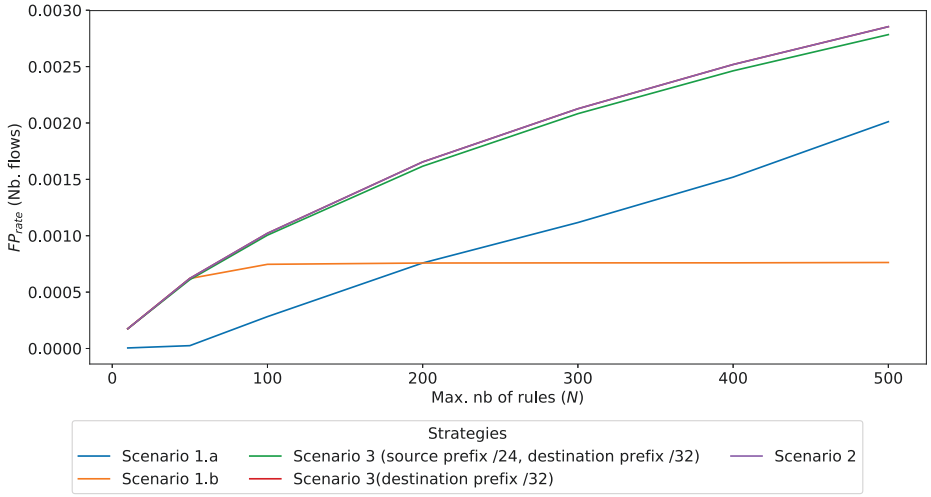
In order to consistently run experiments with the MAWI capture and one of the 10 attack traffics, we follow the following procedure. We select 1000 legitimate sources from the MAWI capture that will also send attack traffic. This emulates the fact that, for example, considering amplification attacks, the target legitimately connects with some (1000 in our case) amplifiers, such as DNS servers. This may be the case when the target is a proxy or a NAT gateway. The remaining malicious sources are randomly chosen such that they are not seen in the legitimate capture. In total, 200,000 malicious sources are selected. We generate a constant bit rate attack traffic with a bandwidth of 1.3Gb/s. Each of the MALaggs has also a constant throughput throughout the attack. While the overall number of attack sources is realistic [4, 1], the overall volume falls short of the most massive current attacks due to computational constraints. MALagg bitrates are randomly distributed within 0.6 × average bitrate and 1.4 × average bitrate.

Scores are regularly re-computed (*e.g.* every 60 seconds in Table 4.4) to update the variation of legitimate traffic. Figure 4.9 and Figure 4.10 show the average true and false positive rates (TP_r and FP_r) for each scoring function with multiple *ALs* (respectively 24 and 8) and varying values of N (x-axis).

[9] MAWI capture is available at http://www.fukuda-lab.org/mawilab/v1.1/2017/02/03/20170203.html



(a) TP_r (number of flows)

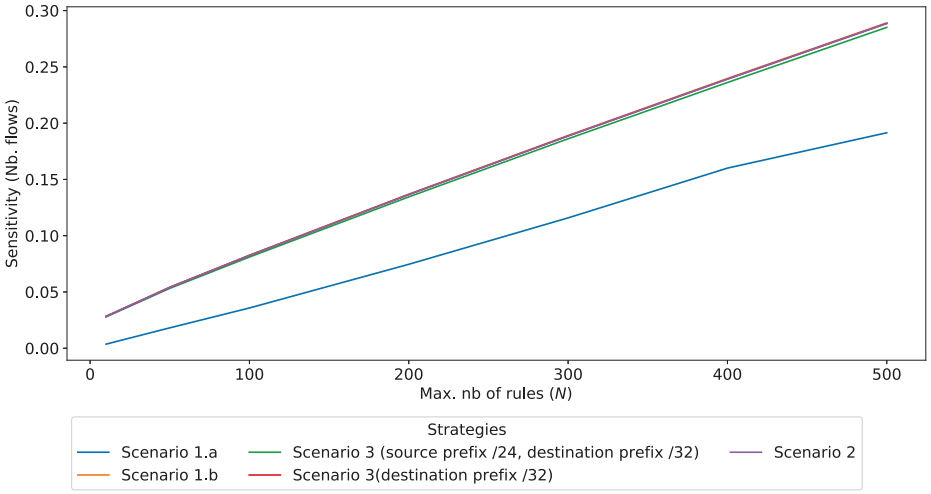


(b) FP_r (number of flows)

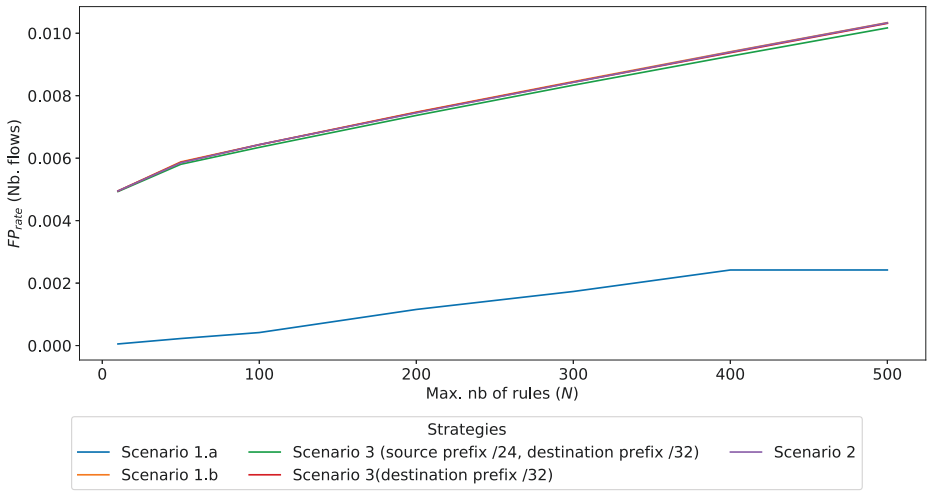
Figure 4.9 – Comparison of scoring functions ($AL = 24$)

Scenario 1 - Minimum Requirements

For *Scenario 1.a* scoring function, the TP_r shows odd trends. For example, in Figure 4.9b, the FP_r has a small increase for N varying from 10 to 50. This growth increases for a number of rules greater than 50. In fact, the strategy does not succeed in selecting the top N rules. This is due to the fact that a lot of filtered aggregates (more than 190,000) have the same score. However, the scoring function has to select the top N, where N is less than 500. There is therefore no rational method to select the top N rules. This results in a pseudo random selection of filtered aggregates.



(a) TP_r (number of flows)



(b) FP_r (number of flows)

Figure 4.10 – Comparison of scoring functions ($AL = 8$)

The *Scenario 1.b* scoring function (in orange) grows linearly from 10 to 50 rules in the filter for each sub-figure. In fact, the number of malicious IPf added per rule linearly decreases considering a maximum rule count greater than 50. However, the TP_r of the *Scenario 1.b* scoring function shows a much larger increase for $N = 10$ to 50. This is also due to the distribution of traffic, where the top N rules contain very dense malicious aggregates with significantly high volumetry. The *Scenario 1.b* scoring function is then correctly choosing the top 10 to 50 rules. This large increase for small N values does not appear in Figure 4.10a, as malicious sources are more evenly

distributed among the source aggregates for small prefixes. In other words, in Figure 4.10a, traffic aggregated in /24 source prefixes emphasizes some aggregates with high impact. However, these /24 aggregates with high score are diluted in /8 aggregates.

Scenario 2 - Enhanced Detection

The TP_r and FP_r of the *Scenario 2* scoring function (in violet) coincides with the *Scenario 1.b* from $N = 10$ to 50 in Figure 4.9. Using malicious IPf telemetries provide no added value given a small number of rules in a filter. Conversely, from $N = 50$ to 500 in Figure 4.9a, the TP_r of the *Scenario 2* grows faster than for the baseline scenario. For example, given $N = 500$ (*cf.*, Table 4.4), the number of dropped malicious flows in *Scenario 2* is just over twice the amount for the *Scenario 1.b*, and the same proportion applies considering the filtered volumetry. The drawback is that the FP_r also grows faster (*cf.*, Figure 4.9b), resulting in an increase of the collateral damages. Table 4.4 shows that, for $N = 500$, the legitimate traffic suffers from filtering around 4 and 5 times more in *Scenario 2* than in *Scenario 1.b*, both in terms of number of flows and volumetry. Both behaviors are mostly due to the fact that the scores of *Scenario 1.b* decrease when the source prefix of the MALaggs grows, that also induces an increase of the probability to include legitimate traffic. The *Scenario 2* does not try to reduce the collateral damages. However, the chosen scenario scoring function is not so efficient with an *AL* of /24, as only 1.5% of malicious traffic is filtered.

When we shorten the *AL*, *e.g.*, from Figure 4.9 to Figure 4.10, the *Scenarios 2* and *1.b* display similar trends. In fact, a large part of malicious IP flows is quite dispersed, so that long source prefixes only aggregate a small part of malicious traffic. As a consequence, in Figure 4.9a, the *Scenario 1.b* scoring function favors malicious flows over malicious aggregates. In contrast, shorter source prefixes (*i.e.* up to /8) aggregate more malicious traffic, so that they get a higher score than longer prefix aggregates. As a consequence, filtered aggregates selected by the *Scenario 1.b* scoring function match the ones selected by the *Scenario 2* and the FP_r curves coincide, *cf.*, Figure 4.10a. Both *Scenarios 1.b* and *2* scoring functions allow filtering around 30% of malicious traffic (in terms of number of flows and volumetry), using solely 500 rules with an average of 1% of filtered legitimate traffic (Table 4.4).

Scenario 3

The curves of *Scenario 3* scoring function configured with a /32 destination prefix granularity (depicted in red) coincide for all sub-figures with the results of *Scenario 2*. Given an *AL* of /24 (Figure 4.9a), this is due to the fact that very few filtered aggregates contain both legitimate and malicious traf-

Table 4.4 – Average statistics of dropped traffic for each scenario considering a blacklist generation each 60s ($N = 500$)

			Scenario 1.a	Scenario 1.b	Scenario 2	Scenario 3 (source prefix /24, destination prefix /32)	Scenario 3 (destination prefix /32)
$AL = /24$	malicious traffic	#IP flows	2222 (1.11%)	1234 (0.62%)	3136 (1.57%)	3107 (1.55%)	3136 (1.57%)
		std	50	55	25	28	25
		MBytes	108.5 (1.11%)	60.28 (0.62%)	154.89 (1.59%)	153.68 (1.58%)	154.89 (1.59%)
		std	2.63	2.81	1.38	1.47	1.38
	legitimate traffic	#IP flows	1222 (0.2%)	462 (0.08%)	1730 (0.29%)	1687 (0.28%)	1729 (0.29%)
		std	814	266	1010	2012	1010
		MBytes	8.06 (0.37%)	2.89 (0.13%)	14.68 (0.67%)	10.45 (0.48%)	14.67 (0.67%)
		std	1.73	0.91	3.42	2.69	3.42
$AL = /8$	malicious traffic	#IP flows	38291 (19.15%)	57822 (28.91%)	57722 (28.86%)	57020 (28.51%)	57713 (28.86%)
		std	164	168	172	226	171
		MBytes	1866.1 (19.14%)	2819.38 (28.92%)	2824.54 (28.97%)	2790.21 (28.62%)	2824.02 (28.96%)
		std	8.34	8.67	8.48	11.18	8.46
	legitimate traffic	#IP flows	1437 (0.24%)	6195 (1.03%)	6191 (1.03%)	6096 (1.02%)	6182 (1.03%)
		std	29	1657	1656	1665	1657
		MBytes	0.22 (0.01%)	32.16 (1.47%)	32.16 (1.47%)	32.02 (1.46%)	31.62 (1.45%)
		std	0.01	5.25	5.23	5.03	5.27

fic, so that introducing monitoring information has not provided much added value. For an *AL* of /8, (*cf.* Figure 4.9b), the reason of the curve adjacency is that legitimate and malicious traffics are highly distributed among filtered aggregates, so that scoring functions get similar results for the rules. This also explains the fact that the FP_r trends of *Scenario 3* configured with < `/24 source prefix, /32 destination prefix` > granularity (shown in green) results almost similar to the *Scenario 2* false positive rate. While this means that it does not reduce the efficiency of the blacklist in terms of malicious traffic filtering, this does not help in preserving legitimate traffic.

The *Scenario 3* that uses < `/24 source prefix, /32 destination prefix` > MONagg granularity, depicts an improvement of the FP_r compared to the *Scenario 2* in Figure 4.10a. Although this seems small when expressed in terms of number of flows (*i.e.*, 0.01% of legitimate flows preserved, *cf.* Table 4.4), results are a little more significant when the FP_r is expressed in terms of volume (0.2%).

4.5 Discussion

4.5.1 Results

Experiments with attack traffic without variations show the basic efficiency and behavior of scoring functions for the considered scenarios. When, considering the hardware limitations of the number of rules in a router, the *Scenario 1.a* scoring function is irrelevant. In fact, the strategy does not allow choosing correctly the top N rules, as more than N rules obtain the best score. We found that with our sets of MALagg, in order to use it effectively, routers would require at least around 190,000 rules for an *AL* of /24. This minimum number of rules is dependent on the malicious traffic distribution, *i.e.*, poorly distributed malicious traffic would require fewer rules to be aggregated.

Scenario 2 reaches the highest efficiency when it comes to only filtering malicious traffic. However, as we increase the *AL*, the *Scenario 1.b* scores results similar to the *Scenario 2* ones. In other words, the optimal efficiency in the *Scenario 1* is at the same level as the efficiency of a scenario with a higher level of detail (*Scenario 2*), assessed in terms of the number of dropped flows. This is a specific case induced by the random distribution of per flow throughput.

Scenario 3 depicts the addition of flow records extracted from the monitoring system. Of particular interest is the coincidence of the false and true positive rates of functions $score_2$ and $score_3$, where $score_3$ has been fed with coarse grained flow records, *i.e.* related to destination IP addresses. This highlights the fact that using coarse grained monitoring information does not increase the efficiency of filtering, in terms of both malicious traffic filtering or legitimate traffic preservation. Conversely, the $score_3$ function

configured with finer grained flow records shows a reduction in the amount of collateral damages, accompanied to a lesser extent by a reduction of the amount of filtered attack traffic. In fact, handling the differing granularities of alert and flow records is harder with a coarse grained granularity of the flow records.

To summarize, experiments demonstrate the impact of the visibility on the efficiency of blacklists. This also highlights the fact that having access to more information, *e.g.* from the monitoring system, does not necessarily induce a more efficient filtering, especially when information (flow records) are coarse grained which leads to a wrong estimation of legitimate traffic inside filtered aggregates. We provide a formalization of the visibility context for studying blacklist efficiency. Using that context model to test blacklists allows comparing of generation algorithms.

4.5.2 Dynamic Attack Traffic

We focused in this part on IPf with constant throughput. We know that attack traffic sources appear and disappear over time. Several reasons account for this, but the main one is to evade detection and static filtering. According to Mirkovic *et al.* [6], DDoS sources can be either persistent, *i.e.*, the attacker uses a single pool of machines whose IPs are present during the whole attack, or intermittent. In the latter case, the attacker controls multiple pools of sources (*e.g.* in botnet-based attacks) or uses several pools of amplifiers which are in turn enlisted in the attack. We discuss this case thereafter.

We evoked the attack traffic composition in Subsection 4.1.1, but we did not address the attack traffic dynamic as it depends on the attacker. Literature classifies attack throughput variations in multiple categories [8, 6]. A *constant rate* describes an attack for which either the volume or the throughput is constant (*cf.*, Subsection 4.4.2). *Pulsing* attack shows peak, while *fluctuating* reflects attack which increases or decreases. As described in Subsection 4.4.1, real attack traffic sometimes shows such pulsing dynamic.

We expect that extending the study to dynamic attack traffic would highlight new contextual parameters. These new parameters would affect the efficiency of blacklist, namely:

- the monitoring collection period, *i.e.* the time between two consecutive exports of flow records to a collection server by the capturing equipment, *e.g.* routers. It denotes the ability of the mechanism to adapt to traffic changes, and especially legitimate traffic changes. A short period is costly, in terms of bandwidth consumed by the flow collection from equipment by the monitoring system. It allows a better adaptation to the legitimate traffic dynamic but is prone to errors caused by traffic noise. For example, short burst of traffic increases the

volume of a flow record which appears as significant at a time t, but are no longer being significant at a time $t+1$. However, flow records related to time t are taken into account at time $t+1$. Longer period is less sensitive to traffic noise, but such errors will affect the filtering over a longer period. The tune of monitoring collection period can be compared to the discussion about flow sampling [168].

- the time between two consecutive alerts which updates identification of malicious traffic. However, some detection mechanisms do not send alert updates. This period depicts the responsiveness of the mitigation. In fact, a short period allows to quickly include new malicious flows in the blacklist. Conversely, a longer period increases the risk of network congestion as new high bandwidth flows are likely to be unfiltered for a long period and then consume bandwidth.

- the blacklist refresh period, *i.e.* the time between consecutive reconfiguration of the blacklist in order to keep an up-to-date mitigation. This period is constrained by the minimum of the two previous periods. In fact, it is unnecessary to refresh blacklists when no change to both malicious and overall traffic has been reported.

Blacklist generation algorithms require the refinement of scoring function following the addition of time-based parameters. As a consequence, history-based algorithm should be studied. For example, Exponentially Weighted Moving Average (EWMA) has already been proposed in [164, 169], to generate blacklists which handle temporal trends of malicious IPf contributions.

4.5.3 Detection Mechanism

In our approach, we assumed that detection mechanisms provide an exhaustive list of attack flows (*i.e.* recall is equal to 100%) where flows cannot contain solely legitimate traffic. In a real world, this is not true in all cases, as the detection mechanisms are not perfect and/or do not report attack sources exhaustively due to the potentially large number of sources in DDoS attacks. However, to be efficient, a detection mechanism is likely to provide top malicious aggregates, *i.e.*, the aggregates with most impact, *e.g.*, the aggregates with the highest data rate as we deal with volumetric DDoS. The malicious aggregates not included in the alert are thus likely to have only a small effect in the network congestion. However, such undetected malicious mice flows are then considered as legitimate.

Table 4.5 shows a qualitative estimation of the potential impact of undetected malicious mice flows on the ability to filter malicious traffic, compared to the ideal situation where the detection is exhaustive. The impact, if any, is limited using generation algorithms that does not take into account monitoring information, *i.e.* in *scenarios 1* and *2*. Nonetheless, the impact

	highly distributed malicious mice flows	localized malicious mice flows
$score_{1.a}$	low impact	Non qualifiable
$score_{1.b}$	low impact	medium impact
$score_2$	low impact	low impact
$score_3$	medium impact	high impact

Table 4.5 – Potential impact of detection mechanism recall on true positive number

depends on the distribution of undetected malicious mice flows. For example, highly distributed mice flows would have less weight (in term of volume of traffic or number of flows) on potential filtered aggregates than localized ones. Localized malicious mice flows has a medium impact on the filtering using the $score_{1.b}$ scoring function, as the function depends on the number of malicious flows inside an aggregate and mice flows can be more numerous than detected malicious flows (elephant flows). In fact, volume of undetected malicious flows only represents a small portion of overall attack traffic volume. As a consequence, mice flows can have a very low throughput, but can also be numerous as long as the sum of their volume is insignificant. When using monitoring information (*scenario 3*), impact is worsen compared to *scenarios 1* and *2* given that malicious flows are considered as legitimate. Finally, the *scenario 2* seems to be the less impacted by the miss detection of mice flows. However, we should confirm these results by experiments.

Chapter 5

Conclusion

Contents

5.1 Contributions

All along this thesis, we pursued the objectives we set at the beginning of this dissertation as regards of the mitigation of volumetric DDoS attacks.

- *Objective A* was to design a mitigation mechanism for volumetric DDoS attacks that benefits from capabilities of existing network equipment.
- *Objective B* was to provide a damage control mechanism which filters, at least, enough attack traffic to prevent network saturation and preserves legitimate traffic.

To meet the *objective A*, we built a Multiprotocol Label Switching-based load-balancing mechanism to isolate suspicious traffic from legitimate. The traffic segregation is achieved with the help of a specially crafted signature which is carried in an MPLS label. Using this Mitigation Label, a load-balancing router is able to deterministically forward malicious and legitimate traffic through a mitigation and a prioritized link, respectively. Malicious traffic which has been isolated on a mitigation path can therefore be treated more restrictively than the legitimate traffic. Using existing network functions, for example provided by the Differentiated Services MPLS

extension, malicious traffic on the mitigation path is forwarded with a lower priority than the legitimate traffic.

We showed that our technique is efficient compared to a common load-balancing scheme. It is able to preserve a large part of legitimate traffic even when the links are saturated by the attack traffic. The use of an MPLS label for load-balancing purpose has already been standardized and implemented in off-the-shelf routers. Although our simple damage control load-balancing function has to be implemented in load-balancing routers, other MPLS routers downstream remain off-the-shelf equipment.

We also study the use of blacklists against volumetric DDoS attacks. The requirements for the equipment deploying the filtering have been carefully reduced. We focused our study to IP-layer blacklists, *i.e.* whose rules are defined only by a source and destination IP prefix, so that we do not impose strong requirements on network equipment that achieve filtering. So, our approach applies to a wide range of equipment.

In response to the *Objective B*, our MPLS-based load-balancing mechanism is based on a signature of the attack. The signature generation process has been designed to meet this goal. The generation process is aware of the volume of malicious and legitimate traffic and the available bandwidth of the prioritized path in order to avoid saturating it. The remaining traffic forwarded through the mitigation path is mostly comprised of malicious traffic and can be rate-limited. Our load-balancing technique showed its ability to preserve legitimate traffic better than similar techniques proposed in the literature while preventing network congestion.

We also provided a study of blacklists, a commonly advertised technique against volumetric DDoS attacks. We especially assessed their ability to filter malicious traffic and the induced collateral damage. Starting from the consideration that network operators have unequal visibility on their networks, we formalized the mitigation context using the visibility concept. We defined four blacklist generation algorithms aimed at maximizing malicious traffic being filtered while minimizing collateral damages. Each of these generation algorithms has different input data requirements reflecting different levels of visibility, namely identification of malicious traffic, per flow volume of malicious traffic, and telemetry about network traffic inside the network. We showed that using more information in blacklist generation does not necessary translate as a better filtering. We propose a first step in formalizing experimental settings for blacklist generation algorithms, in order to ease their comparison.

5.2 Future Work

We split the perspectives for future work in two categories, namely the enhancement of the Mitigation Label load-balancing technique and the decision

aids.

5.2.1 Mitigation Label Enhancement

Our proposed MPLS load-balancing scheme forwards malicious traffic through the mitigation path, but it also forwards a part of legitimate traffic through this link. This creates collateral damage which grows with the attack strength. We highlighted that collateral damages can be worse than the impact of the attack as , for example TCP-based traffic is more impacted by congestion than UDP-based traffic [14]. The forwarding treatment of the mitigation can be refined by prioritizing some part of malicious traffic over the other, so that legitimate TCP traffic ending on the mitigation path suffer less. For example, TCP part of malicious traffic can be prioritized over the UDP part of traffic. More traffic differentiation can even be achieved as for example quality of service mechanisms can drop first TCP SYN packets instead of TCP PUSH or ACK packets. This would then prioritize existing TCP connections by prevent the establishment of new ones. While this traffic sub-categorization can help in case of UDP-based volumetric DDoS attacks, this is not the case against TCP-based attacks. We should note that, the traffic forwarded through the prioritized path take precedence over the part of traffic of the mitigation path with the highest priority.

The QoS sub-classes of malicious traffic can be identified all along the mitigation path using the EXP field of the MPLS header which has been extended for such a use in [113]. Such traffic sub-categorization requires the identification of sub-classes at the network entry point. It increases the consumption of processing resources of ingress Label Switching Routers which requires to achieve a deeper traffic inspection. This is especially true for equipment that are already on pressure from the high volume of traffic implied by volumetric DDoS attacks [118]. The ingress Label Switching Routers CPU consumption can be addressed by implementing routers in commodity hardware [170] or with hardware-accelerated network cards (FPGA) [171].

5.2.2 Decision Support

Although we studied the efficiency of simple blacklist generation algorithms in order to prove the impact of network information availability, there is no comparison basis of existing algorithms with regards of the level of available details we defined in this thesis. Comparing the algorithms proposed by literature based on contextual information will allow an automated choice of the algorithm to react to the attack. In fact, an operator would have to provide his context, *i.e.* attack identification and network information granularities, bandwidth resources, to determine the best mechanism to generate blacklists.

This decision support concept can be extended to the other mitigation

techniques. We provided at the beginning of this dissertation an identification of possible countermeasures (or combination of countermeasures) to volumetric DDoS attacks. The idea is to first model countermeasure capabilities, limits and requirements. Then to build a network context using network topology and location of equipment which has mitigation capabilities. Correlating the context with the countermeasure model and attack details would allow to list suitable mitigation candidates for an attack.

Such decision aid is based on the ability to model the countermeasures, which turns out to be complicated. As we have seen, the mitigation techniques are heterogeneous. For example, some techniques such as blacklists are deterministic, *i.e.* we know which network packet is going to be dropped, while other techniques are probabilistic, such as queuing mechanisms. Surjective reaction techniques depict mechanisms which are based on a reduction of a "signature" such as with hashes. We believe that the granularity and the semantic of mitigation are key concepts of the countermeasure model. The granularity reflects the ability to act on a subset of traffic. The semantic of the reaction, *e.g.* deterministic, probabilistic, surjective or random, express how precisely the mitigation is able to act on attack traffic within the subset of traffic.

5.3 Closing Words

We are given the opportunity to work on a topical issue, that is volumetric DDoS attacks. We have seen much activity around this area during the thesis, especially due to the emergence of new attack vectors, such as Internet of Things botnets, which leverage new peak of volume of traffic. We approach a variety of fields such as Distributed Denial of Service attacks and responses, MPLS forwarding, network traffic management, congestion control, Quality of Service and User Experience evaluation. We aimed at building a volumetric DDoS attack mitigation mechanism. We proposed a load-balancing scheme to segregate malicious traffic from legitimate traffic and forward each through a dedicated path, so that malicious traffic can be constrained. We also studied the efficiency of blacklists, a commonly used mitigation technique, in regard of operational constraints such as availability of information about the protected network.

Appendix A

French Summary

A.1 État de l'art

Dans le cadre des technologies de l'Information, Internet est soumis aux mêmes critères fondamentaux d'évaluation, à savoir, la *confidentialité*, l'*intégrité* et la *disponibilité*. La confidentialité caractérise le fait qu'un individu non autorisé n'ait pas accès à une donnée. L'intégrité décrit la capacité à prévenir la corruption ou modification non autorisée d'une donnée. La disponibilité reflète la capacité pour un utilisateur à accéder à une donnée quand il en a besoin. Dans ce contexte, les attaques par déni de service (*DoS*) ciblent particulièrement, et uniquement le critère de disponibilité en perturbant un service fourni sur Internet. Les préoccupations relatives aux attaques DoS ont considérablement augmenté au fil des années, suite à une augmentation de leur nombre, fréquence, taille et de leur criticité. De plus, les connaissances nécessaires au lancement de telles attaques ont diminué. Par exemple, le volume des attaques n'a cessé d'augmenter ces dernières années, le record passant de 300 Gbps en 2013 [3] à plus de 1 Tbps en 2016 [1]. En 2016, l'émergence de réseaux de machines zombies (machines contrôlées par un attaquant) a ouvert la voie à des attaques encore plus massives. Au-delà de l'interruption du service ciblé, cela met en danger de multiples acteurs de l'Internet comme des fournisseurs de service ou des centres de données. C'est le cas, par exemple, de l'attaque visant les serveurs DNS de Dyn en 2016, touchant des sites de renoms tels que Netflix, Amazon, Twitter. Cependant, même si la plupart des attaques ont un volume assez faible (500 Mbps selon une étude d'Arbor Networks [5]) comparé au record de 1 Tbps, elles menacent la disponibilité de cibles non protégées ou insuffisamment protégées.

Il est important de noter que ces valeurs de volume record se rapportent à un débit cumulé sur les différents points de présence de la cible sur Internet. La force de frappe serait encore plus dévastatrice si l'attaque était concentrée en un seul point.

Caractérisation des attaques volumétriques par déni de service distribué Dans cette dissertation, nous nous concentrons sur les attaques volumétriques par déni de service distribué (DDoS). La Figure A.1 présente une infrastructure réseau touchée par une telle attaque. Les clients légitimes effectuent des requêtes, représentées en vert, vers un service sur Internet. En rouge, des sources distribuées sur les points d'entrée du réseau envoient du trafic réseau à destination de ce service. Dans le cas d'une attaque à faible volume, l'infrastructure réseau et une solution anti-DDoS supportent la charge. Ainsi la solution anti-DDoS nettoie le trafic d'attaque et renvoie les requêtes légitimes vers le service. Une attaque volumétrique réalise le déni de service en saturant la bande passante de liens réseau, congestionnant par conséquent le réseau. Cependant des attaques massives peuvent tout aussi bien surcharger des équipements sous dimensionnés, qui ne peuvent opérer à la vitesse du lien réseau. Parmi ces équipements, on note les solutions de sécurité ou encore des ressources situées en amont comme des routeurs ou des pare-feu. L'épuisement des ressources est en grande partie due à l'effet de convergence qui concentre le trafic réseau provenant de sources dispersées vers une même cible, comme le montre la Figure A.1. Le fait que l'engorgement se produise en amont de la solution de mitigation dédiée la rend inefficace puisque les requêtes légitimes, noyées parmi le trafic d'attaque, sont déjà perdues des suites de la congestion. Le service internet est ainsi rendu indisponible, totalement ou partiellement, pour les clients légitimes.

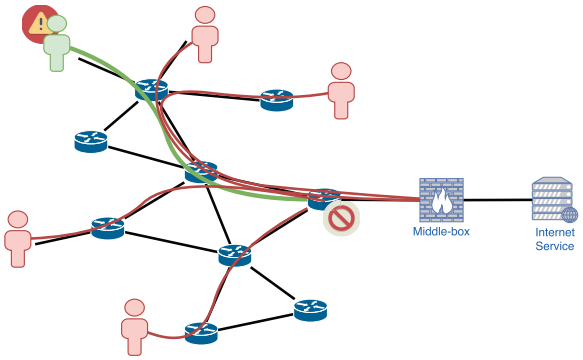


FIGURE A.1 – Example d'attaque par déni de service distribué volumétrique contre un service Internet, causant la saturation d'un lien réseau

Comme une partie du trafic réseau est encore acheminée jusqu'à la cible, quelques requêtes légitimes peuvent encore atteindre la cible. Selon le type de service fourni, la perception de l'indisponibilité par l'utilisateur légitime peut varier. On parle alors de qualité d'expérience utilisateur. Par exemple, la perte de paquets réseau pour un service de vidéo basé sur le protocole UDP peut seulement résulter en une qualité de vidéo dégradée ou un temps de chargement allongé. À l'inverse, d'autres services basés sur le protocole TCP, peuvent maintenir les équipements réseaux sous pression à cause des

mécanismes de retransmission inhérents au protocole.

De plus, les attaques par déni de service distribué exploitent la dispersion des sources d'attaques pour générer une grande quantité de trafic convergeant vers une même destination. Un attaquant a à sa disposition plusieurs moyens pour générer du trafic depuis un grand nombre de sources.

Les réseaux de machines robots (botnets) sont composés d'équipements (serveurs, ordinateurs personnels, appareil de l'Internet des Objets) qui peuvent avoir été infectés ou enrôlés volontairement [10]. Par conséquent, l'attaquant a la capacité de coordonner une attaque depuis ce groupe de machines. Les robots envoient du trafic directement à la cible pouvant générer des attaques requérant du trafic bidirectionnel. Aussi du point de vue de la victime, le nombre de sources vues est contraint par le nombre d'équipements qu'un attaquant est capable de contrôler. Un tel nombre de machine rend possible des attaques d'inondation de requêtes applicatives, comme des requêtes HTTP de type POST, contenant un fichier à téléverser.

Cependant, pour éviter d'être détecté ou filtré, les robots peuvent envoyer du trafic usurpé en exploitant des protocoles en mode sans connexion. Dans ce cas, les robots envoient du trafic dont l'adresse IP source est falsifiée, induisant pour la victime un nombre de sources vues potentiellement infini. Nous ne considérons pas ce type d'attaque dans notre dissertation.

Les attaques DDoS par amplification, et plus généralement par réflexion, exploitent le mode sans connexion des protocoles comme UDP. Une machine effectue une demande à un service, avec pour adresse IP source celle, falsifiée, de la victime. Le service envoie donc à la victime sa réponse. Si la réponse est plus volumineuse que la requête, on parle d'attaque par amplification, et le service trompé est appelé amplificateur. Même si l'attaquant falsifie initialement la source de sa requête, l'adresse IP source relative au trafic de l'amplificateur vers la cible est non falsifiée. Le nombre de sources vues par la cible est donc limité au nombre d'amplificateurs qu'un attaquant peut exploiter. Dans cette dissertation, nous nous positionnons entre l'amplificateur et la cible.

Dans les sections suivantes, nous classifions les mécanismes de limitation des attaques par déni de services suivant trois catégories. Les techniques de contrôle d'admission déterminent quelle partie du trafic doit ou non entrer ou continuer de traverser le réseau. Les techniques basées sur les déviations de chemins sont souvent utilisées avec d'autres types de mécanismes. Les techniques d'absorption maintiennent la disponibilité du réseau en atténuant ou en supprimant les problèmes de congestion, et cela en utilisant les ressources disponibles.

A.1.1 Contrôle d'admission

Les mécanismes de contrôle d'admission déterminent si une requête à un service doit être transmise ou bloquée. Ces mécanismes peuvent nécessiter une modification de la communication entre le client et le service pour prendre sa décision (mécanisme passif), ou non (mécanisme passif).

Contrôle d'admission actif

Cette catégorie regroupe deux techniques principales. La première, basée sur des défis, nécessitent que les clients prouvent leur légitimité en répondant à un défi. L'idée est qu'un attaquant n'a pas les ressources en termes de puissance de calcul ou d'intelligence (par exemple pour des défis visuels) pour relever les défis à grande échelle. Cela repose, cependant, sur l'ajout d'une étape lors de la fourniture du service, puisque les clients doivent répondre au défi. Ce principe n'est donc pas applicable à tous les types de service, notamment à ceux qui requièrent une très haute disponibilité, comme entre autres la résolution d'un nom de domaine. En effet, dans ce cas, l'ensemble de l'architecture de l'Internet souffrirait d'un tel manque de spontanéité dans la réponse à une requête.

La seconde technique, basée sur la notion de capacité, fait l'hypothèse qu'un serveur peut discriminer le trafic d'attaque du trafic légitime. Celui-ci ajoute donc avec sa réponse à une requête, son approbation (*i.e.* capacité) à continuer de recevoir du trafic de ce client. Cependant, essentiellement dans le cas des attaques par déni de service volumétrique, la saturation se produit en amont du serveur, de telle sorte que le trafic légitime est déjà perdu parmi les requêtes malicieuses. Il ne peut donc pas obtenir l'assentiment du serveur.

Contrôle d'admission passif

Les mécanismes de contrôle d'admission passif sont regroupés en six sous catégories : le filtrage, l'ordonnancement, la gestion des queues de transfert, la limitation du trafic, la distribution de mécanismes de filtrage et la combinaison de plusieurs mécanismes.

La capacité d'un mécanisme de filtrage dépend de la granularité à laquelle il peut agir. Un mécanisme à grosse maille filtre avec succès le trafic malicieux, mais induira en contrepartie des dégâts collatéraux importants. À l'inverse des mécanismes plus fins, intégrant par exemple du suivi de connexion ou de l'inspection en profondeur de paquets réseaux, sont capables de mieux différencier le trafic légitime du trafic d'attaque, réduisant ainsi les potentiels dégâts collatéraux. Des mécanismes aussi fins, ne peuvent cependant pas être intégrés dans des équipements ordinaires et demandent plus de ressources que des mécanismes de filtrage à grosses mailles. Ils sont, par conséquent, souvent regroupés en bastion, requérant ainsi de rediriger le trafic vers ceux-ci.

Les mécanismes basés sur l'ordonnancement ont été introduits initialement pour prioriser le trafic. Leur usage a été modifié pour leur permettre de renforcer l'équité entre les différents flux composant le trafic. Ces mécanismes sont distribués dans les routeurs permettant ainsi d'agir en amont du point de saturation causé par une attaque. Cependant renforcer l'équité entre les flux n'est pas suffisant dans le cas d'une attaque par déni de service volumétrique. En effet, le mécanisme octroie dans ce cas, une part de bande passante équitable, même aux flux malicieux, puisqu'il traite indifféremment les trafics légitime et malicieux. Le trafic malicieux continuant de consommer de la bande passante, un nombre grandissant du nombre de flux d'attaque induira une réduction de la bande passant équitable.

Les mécanismes basés sur les queues tentent de prévenir la congestion en agissant de manière proactive. Cependant dans le cas d'une attaque par déni de service ils agissent de manière réactive, puisqu'ils cherchent à partager équitablement la bande passante entre tous les flux du trafic. Cela implique que pendant un cours instant chaque flux pourra envoyer à un débit atteignant le débit équitable, causant ainsi le même problème que pour les mécanismes basés sur l'ordonnancement.

Les mécanismes de limitation du trafic contraignent le trafic (ou une partie du trafic) à une certaine bande passante. Ces mécanismes ne permettent pas par eux même de spécifier sur quelle partie du trafic ils s'appliquent. Par conséquent, ils doivent être combinés avec d'autres mécanismes. L'efficacité de la combinaison dépendra de la capacité à restreindre le trafic sur lequel s'applique la contrainte afin de ne pas permettre au trafic malicieux de consommer toute la bande passante.

La distribution des mécanismes de filtrage à grosse maille permet d'éviter l'effet entonnoir du trafic d'une attaque par déni de service. Cela augmente donc la résilience du réseau à ce type d'attaque et permet aussi une meilleure gestion des ressources puisque le filtrage peut ainsi dépendre de la localisation du trafic d'attaque dans le réseau. Cependant, comme pour tout mécanisme de filtrage, l'efficacité dépend de la granularité à laquelle le trafic est filtré.

Les mécanismes combinés intègrent une détection automatique des parties du trafic à filtrer dans des équipements réseaux comme les routeurs. Ils sont spécialement conçus pour consommer des ressources raisonnables (en termes de puissance calcul) sur ces équipements. Certains mécanismes intègrent aussi des fonctions de coopération entre équipements. Le manque de validation externe ainsi que la granularité importante du filtrage induite par les optimisations de ressources, peut cependant causer d'important dégâts collatéraux.

A.1.2 Techniques basées sur la modification de chemins réseaux

De manière générale, les techniques basées sur la modification de chemins réseaux doit être combinée avec d'autres mécanismes, comme des filtres ou de la limitation de bande passante.

Les mécanismes permettant une optimisation des liens réseaux, par exemple le routage multi-chemin, visent à utiliser l'ensemble des ressources disponibles au sein du réseau, au lieu de les laisser au repos. Cependant, lors d'une attaque par déni de service, le trafic d'attaque pourrait consommer toute la bande passante du réseau. Si cela est peu probable [118], elle peut tout au moins saturer les liens d'entrée, rendant ainsi l'optimisation sans effet.

A.1.3 Techniques d'absorption

Les mécanismes d'absorption basé sur le surdimensionnement font l'hypothèse que l'attaquant n'a pas assez de ressources pour surcharger un réseau qui aura été surdimensionné. Toutefois, dans le contexte des attaques par déni de service volumétriques, une telle technique ne fait que déplacer le point de saturation en amont [118]. Pour éviter ce phénomène, ce type de technique doit être combiné avec d'autre mécanismes comme des filtres.

Les mécanismes de reconfiguration de la pile TCP du serveur ne sont pas applicables à des attaques exploitant le protocole UDP. De plus, ces mécanismes visent essentiellement à protéger d'attaques par inondation de demande de connexions TCP incomplètes (par exemple, les attaques dites SYN floods). Ceci les rends donc inefficaces contre des attaques par inondation de requêtes applicatives utilisant le protocole TCP, comme les requêtes de type HTTP GET. En effet, dans ce cas les demandes de connexion sont achevées par l'attaquant afin d'effectuer la requête applicative.

A.1.4 Conclusion

Les techniques de routage multi-chemin semblent être des techniques prometteuses de contrôle de dommage contre les attaques par déni de service distribué volumétriques. Il peut être combiné avec des mécanismes de différenciation du traitement de transmission en fonction du type de trafic. Le but étant de prioriser la transmission du trafic légitime afin qu'il ne subisse plus la saturation des liens du réseau. Cela nécessite cependant une isolation du trafic malicieux pour qu'un traitement différencié puisse être appliqué.

A.2 Technique de mitigation basée sur le partage de charge

Le routage sur des chemins multiples et en particuliers les mécanismes de partage de charge sont communément implémentés dans des équipements réseaux. Ils sont destinés à mieux utiliser les ressources réseaux, sans coût additionnel comme ceux induits par des fonctions de filtrage. En contrepartie, le routage sur de multiples chemins ne peut de lui-même différencier le trafic légitime du trafic malicieux, nécessitant ainsi une combinaison avec un outil permettant la séparation du trafic afin d'effectuer un traitement différent à chaque type de trafic. Dans cette partie nous étudions la possibilité d'utiliser des mécanismes de partage de charge afin de séparer le trafic malicieux du trafic légitime afin de s'affranchir du coût en ressources des fonctions de filtrages. Devant l'impossibilité de remplir ce but à l'aide des mécanismes existant, nous proposons une technique de partage de charge pour le protocole MPLS.

A.2.1 Partage de charge avec un routeur Cisco

Nous étudions, dans cette section, la possibilité d'utiliser les mécanismes de partage de charge embarqué dans un routeur Cisco 7200 pour isoler le trafic malicieux.

Le routeur Cisco embarque deux algorithmes de partage de charge. Le premier, dit original, se base sur les adresses IP source et destination d'un paquet réseau pour définir le prochain saut vers lequel il sera transmis. Le second, nommé universel, utilise en plus un grain de sel, permettant ainsi à chaque routeur, avec un grain de sel différent, d'obtenir des résultats différents. Nous faisons l'hypothèse que l'algorithme universel est dérivé de l'algorithme original. Dans un premier temps, nous effectuons donc la rétro-ingénierie de l'algorithme original. Le but est de découvrir l'algorithme universel afin de vérifier sa capacité à séparer le trafic légitime du trafic malicieux, en faisant varier le grain de sel. Les deux algorithmes fonctionnent sur le même principe : un hachage des paramètres d'entrée est réalisé puis chaque résultat est assigné à un chemin à l'aide d'une table de correspondance. Cette table, générée à partir des chemins de sortie, est connue. Il reste donc à trouver la définition de la fonction de hachage pour avoir l'algorithme entier.

Nous avons réussi à retrouver manuellement la fonction de hachage de l'algorithme original (*cf.* Equation (A.3)) en observant ses résultats en fonctions des différents paramètres d'entrée (adresses IP source et destination). Les adresses IP source et destination sont notées $s_0.s_1.s_2.s_3$ et $d_0.d_1.d_2.d_3$. La fonction rotleft désigne la fonction qui fait tourner les bits à gauche du nombre de positions donné en paramètre.

$$hash(s_0.s_1.s_2.s_3, d_0.d_1.d_2.d_3) = Hsrc(s_0.s_1.s_2.s_3) \wedge Hdst(d_0.d_1.d_2.d_3) \tag{A.1}$$

$$Hsrc(s_0.s_1.s_2.s_3) = ((s_3 \wedge s_4) \gg 1) \ \& \ 0\text{xF} \tag{A.2}$$

$$Hdst(d_0.d_1.d_2.d_3) = (((d_3 \ll 1) \wedge \text{rotleft}(d_4, 1)) \ \& \ 0\text{xF}) \wedge (s_3 \gg 4) \tag{A.3}$$

Nous n'avons pas réussi à en déduire la fonction de hachage utilisée par l'algorithme universel. Cependant, le gain potentiel de l'algorithme universel est limité. En effet, le grain de sel, sur lequel on comptait pour séparer le trafic légitime du trafic malicieux, est configuré pour chaque routeur. Ainsi, pour mettre en place une protection contre plusieurs attaques, nous devrons trouver un grain de sel convenant à plusieurs contextes : 1) chaque attaque a ses propres couples d'adresses IP source et destination ; 2) les cibles sont situées à différents endroits. Aussi, chaque attaque peut avoir un ensemble différent de chemins. Nous abandonnons donc l'idée d'utiliser de tels algorithmes de partages de charge.

A.2.2 Partage de charge et protocole MPLS

Notre but est de rediriger le trafic malicieux et légitimes par des chemins auxquels on applique des traitements de qualité de service non prioritaires et prioritaires. La solution HADEGA [13] semble un socle prometteur pour l'établissement des chemins et la priorisation. Cette section, examine les mécanismes de partage de charge pour isoler le trafic malicieux du trafic légitime.

Par défaut, les routeurs MPLS se basent sur des informations du protocole IP pour déterminer quel chemin de sortie sera emprunté par un paquet réseau [134]. Diverses extensions ont été ajoutées au protocole MPLS, permettant aussi d'utiliser le label MPLS comme paramètre d'entrée [136, 137]. La fonction de hachage Cyclic Redundancy Check sur 16 bits (CRC16) est citée dans différents papiers [108, 139] et brevets [140, 141, 142] comme fonction de base pour effectuer du partage de charge. Dans de tels algorithmes, les paramètres d'entrée sont hachés et le résultat est assigné à un chemin de sortie. La méthode utilisée pour sélectionner le chemin de sortie en fonction du résultat de la fonction de hachage dépend de l'implémentation par les constructeurs.

Comme le montre la Figure A.2, extraite de [108]), un partage de charge pondéré sur N chemins est effectué en utilisant une table de hachage. Nous utiliserons cette méthode d'assignation des chemins de sortie en ne considérant que deux chemins, à savoir : le chemin dédié au trafic légitime et celui dédié au trafic malicieux.

Nous allons étudier la fonction CRC16 dans le cadre d'un partage de charge se basant sur les paramètres d'entrée suivants : l'adresse IP source

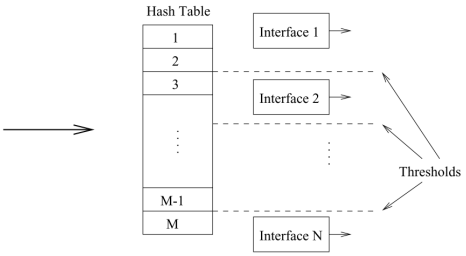


FIGURE A.2 – Assignation du prochain saut, basé sur un table de hachage [108]

ainsi que le label MPLS. Notre but est de trouver un label qui maximise le nombre de couples `< IP source, IP destination >` malicieux transmis sur le lien dédié au trafic malicieux. Nous résumons ainsi notre approche expérimentale. Nous générons d'abord aléatoirement 2000 adresses IP sources malicieuses. Puis nous simulons le mécanisme de partage de charge pour tous les labels MPLS possibles, c'est-à-dire, dont la valeur est comprise entre 16 et 2^{20}, les labels de 0 à 15 étant réservés. Puis, nous comptons le nombre d'adresses IP qui ont été transmises sur le lien malicieux. Nous rejouons cette expérimentation 20 fois, avec différents ensembles d'adresses IP sources. Comme nous avons considéré un partage de charge pondéré, chaque jeu d'expérimentation est répété avec différentes attributions de poids aux liens dédiés au trafic légitime et malicieux.

La Figure A.3 montre la valeur moyenne, minimale et maximale du nombre d'adresses IP sources ayant été transmises sur le lien malicieux (*i.e.* vrais positifs). L'axe des abscisses correspond aux partages de poids testés. Par exemple, "1 : 3" correspond à des poids de 1 et 2 $(3-1)$ assignés respectivement aux liens dédiés au trafic malicieux et légitime. Les barres bleues désignent les résultats expérimentaux, tandis que les barres oranges montrent les résultats probabilistes. En effet, le partage de charge pondéré est un mécanisme probabiliste qui transmet un paquet réseau sur un chemin en proportion du poids qui lui est assigné.

Globalement, pour une certaine répartition des poids, la moyenne du nombre de vrais positifs est proche de la valeur probabiliste. On doit donc noter que pour toutes les répartitions, le nombre minimum de vrais positifs est en deçà de cette valeur probabiliste. Cela souligne le fait que, au moins pour une des expérimentations, le mécanisme de partage de charge n'a pas été capable de transmettre plus d'adresses IP sur le lien malicieux que la valeur proportionnelle aux poids des liens. En d'autres termes, le mécanisme n'a apporté aucune valeur ajoutée comparé à un mécanisme pris sur l'étagère.

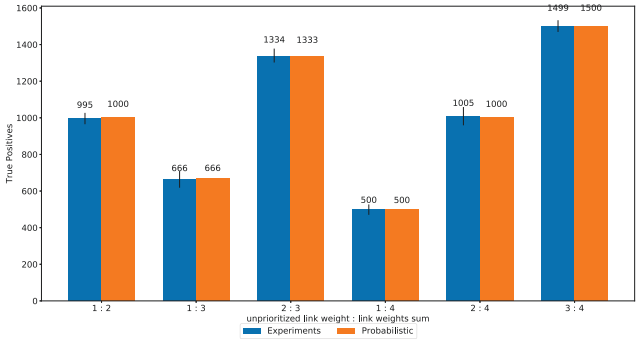


FIGURE A.3 – Nombres de vrais positifs induit par un partage de charge basé sur la fonction CRC16

A.2.3 Mitigation Label : un mécanisme de mitigation des attaques par déni de service utilisant le MPLS

L'architecture de notre mécanisme de mitigation est schématisée dans les Figures A.4 à A.6. Un système de surveillance collecte en permanence des télémétries sur le réseau, comme montré dans la Figure A.4. Le système de surveillance est aussi alerté en cas d'attaque par déni de service. Il peut les détecter par lui-même ou recevoir des alertes de capteurs dédiés, par exemple. Le mécanisme d'atténuation est déclenché par le mécanisme de

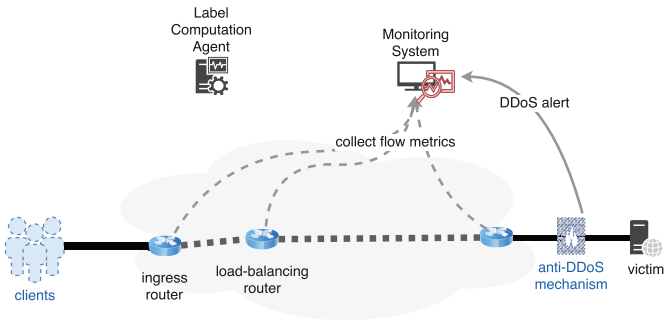


FIGURE A.4 – Détection de l'attaque

surveillance qui transfère l'alerte à un agent de calcul de label (Label Computation Agent - LCA). L'agent traite l'alerte et peut demander des informations (télémétries) complémentaires au système de surveillance. Pour chaque routeur d'entrée (*iLSR*), l'agent génère enfin le label de mitigation qui contient les paramètres d'entrée nécessaires au mécanisme de partage de charge. Pour des raisons de clarté, un seul routeur d'entrée est représenté sur la Figure A.5. L'agent annonce la présence d'un Label de Mitigation à chaque routeur d'entrée qui va le placer sur la pile de label MPLS des

paquets réseaux appartenant à une partie du trafic contenant les flux malicieux. Le Label de Mitigation est donc transmit avec ses paquets réseaux jusqu'au routeur effectuant le partage de charge (Load Balancing Router - LBR). La Figure A.6 schématise le partage de charge effectué par le routeur

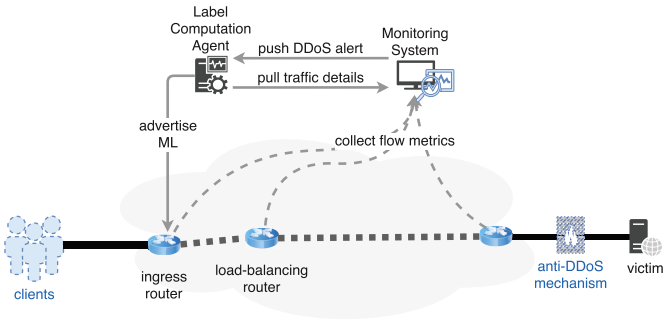


FIGURE A.5 – Génération de la signature de l'attaque et configuration de la contre-mesure

vers deux chemins (LSPs), à savoir le chemin prioritaire et le chemin de mitigation. Le chemin prioritaire, au travers duquel sera acheminé le trafic légitime, devra être préservé au maximum de la congestion. Le trafic acheminé à travers le chemin de mitigation pourra être éliminé par le réseau en cas de congestion.

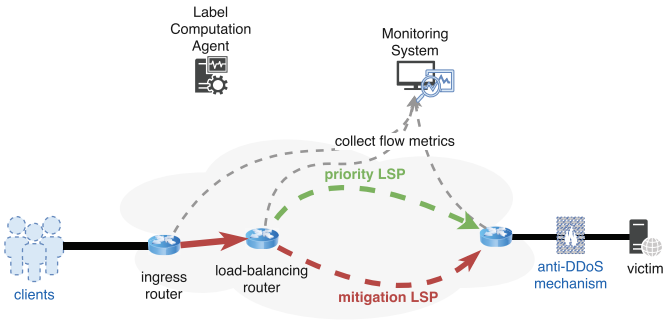


FIGURE A.6 – Atténuation de l'attaque

Données d'entrées Les alertes pour attaques DDoS volumétriques contiennent la caractérisation des composantes d'une attaque en cours. Elles contiennent ainsi l'identification des flux IP suspicieux, représentés par une liste de couple `< adresse IP source, adresse IP destination>` impliqués dans l'attaque. Des formats d'échange d'informations, comme IDMEF [144] ou le format de signalisation d'attaques DDos proposé par le groupe de travail IETF DOTS [145], embarquent de telles informations. Notre mécanisme d'atténuation requiert des télémétries sur les flux réseaux, comme

leur volumétrie. Alors que ces télémétries peuvent être transmises avec les alertes pour les flux IP malicieux, nous considérons qu'elles sont aussi disponibles pour les flux légitimes à travers le système de surveillance réseau. Celui-ci peut, en effet, collecter des télémétries en utilisant par exemple NetFlow [146], sFlow [147] ou IPFIX [148]. Les télémétries sont collectées au moins avec la granularité d'un flux IP.

Filtre de Bloom avec seuil Un filtre de Bloom est une structure de données économique en espace mémoire représentant un ensemble d'éléments S. Sa principale fonction est de tester si un élément fait partie de l'ensemble que le filtre représente. Le test peut conclure avec certitude que l'élément testé n'est pas dans l'ensemble S, soit qu'il y est peut être, avec une probabilité d'erreur (nommée probabilité de faux positifs) donnée. Cette probabilité peut être minimisée en ajustant trois paramètres [149] : 1) la taille de la structure de donnée, qui est un tableau de bit de taille mp ; 2) le nombre d'élément dans l'ensemble S représenté par le filtre ; 3) le nombre k de fonction de hachage utilisé pour générer et tester et le filtre. La construction d'un filtre suit le processus suivant. Chaque bit du filtre de Bloom est initialisé à 0. Les fonctions de hachage assignent un élément à un index du tableau. Le filtre est construit en hachant chaque élément de l'ensemble S avec les k fonctions. Le bit à la position correspondant au résultat du hachage est fixé à 1. Tester si un élément est référencé dans le filtre revient à hacher cet élément et vérifier la valeur de chaque bit aux positions données par les résultats des hachages. Si un des bit est égal à 1, l'élément est peut être référencé, sinon, avec certitude il n'est pas dans l'ensemble S.

Les filtres de Bloom à compteurs [150] sont une extension aux filtres de Bloom, pour lesquels, les bits sont remplacés par des compteurs. Un compteur est incrémenté à chaque fois que le résultat du hachage d'un élément de l'ensemble S a donné sa position en résultat.

Nous utiliserons des filtres de Bloom à compteurs avec une seule fonction de hachage, de telle sorte qu'un élément n'est représenté que par une seule position. Lorsque nous insérons un élément (un flux IP), nous ajoutons le volume de ce flux (exprimé en termes d'octets) aux compteurs au lieu de l'incrémenter. Le volume de ce flux IP suspicieux aura été extrait au préalable du système de surveillance. Nous définissons ensuite le concept de sous-ensemble significatif représentant un sous-ensemble de trafic qui devrait être filtré à cause de sa volumétrie. Le seuil t correspond au volume minimal à partir duquel on considère qu'un sous-ensemble de trafic est significatif. Nous générons ensuite un filtre de Bloom en combinant le filtre de Bloom à compteur et le seuil t. Pour chaque compteur, si sa valeur est supérieure ou égale au seuil, le bit du filtre de Bloom (tBF) à la même position sera instancié à 1, sinon à 0, comme représenté sur la Figure A.7.

Le filtre de Bloom avec seuil est généré en fonction des conditions ac-

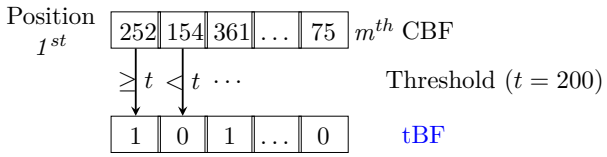


FIGURE A.7 – Exemple de génération d'un filtre de Bloom avec un seuil $t = 200$

tuelles du réseau : le trafic légitime et malicieux ainsi que la capacité du lien saturé par l'attaque. Nous introduisons donc un filtre de Bloom à m compteurs destiné à évaluer l'impact du volume des sous-ensembles de trafic Chaque compteur CBF_{mixed}, à la position i est initialisé comme dépeint dans la Figure A.2.3.

$$CBF_{mixed}[i] = \frac{CBF_S[i]}{1 + CBF_L[i]}, \forall i \in [0; m-1] \ .$$

$CBF_S[i]$ correspond à la somme de trafic des flux IP malicieux qui seront hachés à la position i. $CBF_L[i]$ correspond à la somme de trafic des flux IP légitime qui seront hachés à la position i. Si les flux IP légitimes ne sont pas connus, la somme est nulle. Cette équation permet d'augmenter le poids d'un sous-ensemble de trafic lorsque le volume de trafic malicieux de ce sous-ensemble augmente. À l'inverse l'estimation de l'impact d'un sous-ensemble diminue lorsque le volume de trafic légitime dans ce sous-ensemble augmente.

On considère un tBF déduit de ce filtre de Bloom avec compteurs. On note V_{FP} le volume de trafic légitime effectivement inséré dans le tBF, c'est-à-dire dont le hachage donnera une position dont le bit est à 1. À l'inverse V_{FN} correspond au volume de trafic malicieux dont les éléments n'ont pas été insérés dans le tBF.

$$V_{FP} = \sum_i CBF_L[i], \text{ such that } tBF[i] = 1 \ ,$$
$$V_{FN} = \sum_{\bar{i}} CBF_S[\bar{i}], \text{ such that } tBF[\bar{i}] = 0 \ .$$

La génération du tBF a pour but de prévenir la saturation du lien prioritaire, donc l'allocation en bande passante est notée $A_{priority}$. L'estimation du volume à travers ce lien correspond à la somme du volume des flux dit faussement négatifs (*i.e.* trafic suspicieux sur le lien prioritaire) et vraiment positifs (*i.e.* trafic légitime sur ce même lien). Sachant que le volume de flux vraiment négatif est égal au volume du trafic légitime ($V_{in}leg$) moins la somme des volumes des flux faussement positifs, la contrainte est décrite par l'équation suivante :

$$V_{in}leg - V_{FP} + V_{FN} < A_{\text{priority}} \ . \tag{A.4}$$

La génération du tBF suit la démarche suivante. 1) On initialise le seuil à son maximum (*i.e.* la somme de tous les compteurs CBF_{mixed}. 2) On vérifie la contrainte. 3) Tant que la contrainte n'est pas vérifiée, on décrémente le seuil.

Expériences Nous avons implémenté le mécanisme sous forme d'une plateforme de simulation logicielle. Celle-ci prend en entrée une capture de trafic contenant les composantes légitimes et malicieuses et effectue le partage de charge sur un lien prioritaire et un lien de mitigation. Nous avons effectué une comparaison (visible sur la Figure A.8) entre notre technique de partage de charge (en trait plein bleu) et une technique de partage de charge basique (entrait discontinu rouge). Le gain (*i.e.* différence entre les deux courbes) est tracé avec un trait alterné vert. L'axe des abscisses exprime le ratio entre le volume de trafic malicieux et celui du trafic légitime en entrée de réseau. L'axe des ordonnées est exprimé comme le ratio entre le volume de trafic légitime en sortie et en entrée de réseau.

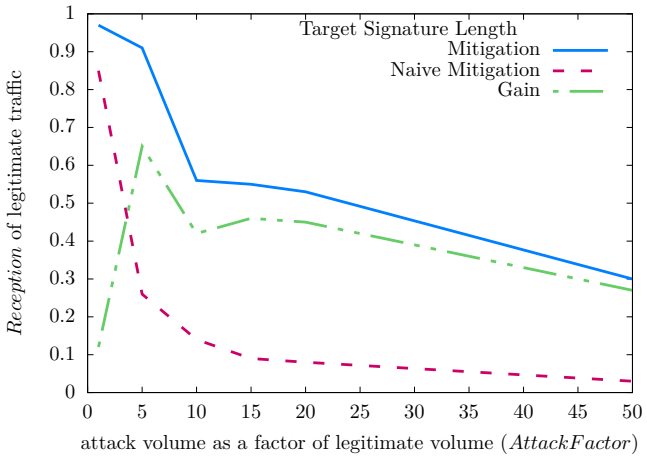


FIGURE A.8 – Comparison of the legitimate traffic reception with and without mitigation

A.2.4 Discussion

Nous avons comparé notre mécanisme avec un mécanisme de partage de charge pondéré, c'est-à-dire que le chemin de sortie est sélectionné en fonction du résultat du hachage d'un paquet, de manière proportionnelle aux poids configurés pour le chemin de mitigation et le chemin prioritaire. En considérant une allocation de bande passante particulière pour le chemin de mitigation, notre solution montre de meilleurs résultats allant jusqu'à un gain de 50% de trafic légitime préservé.

Nous avons aussi considéré l'impact de la bande passante allouée au lien de mitigation sur la réception du trafic légitime en sortie de mécanisme. Les expérimentations ont montré que le nombre de dégâts collatéraux augmentent lorsque l'on augmente l'allocation de la bande passante pour le lien de mitigation tandis que l'allocation des deux liens reste constante. À l'inverse, les dégâts collatéraux diminuent lorsque l'allocation de la bande passante pour le lien de mitigation augmente mais que l'allocation du lien prioritaire reste constante. Cela implique pour un opérateur de réseau, qu'il doit réserver une certaine bande passante pour le lien de mitigation, et ceci par client afin de diminuer les dégâts collatéraux. Cependant le choix de cette bande passante est conditionné par le coût additionnel d'une telle réservation et le coût que causeraient les potentiels dégâts collatéraux.

A.3 Mitigation à base de listes noires

Les listes noires désignent un mécanisme de filtrage pour lequel le trafic correspondant au filtre est rejeté, le trafic restant étant accepté (transmit, routé, ...). Ce type de contre-mesure filtrant sur les adresses IP sources est souvent mis en avant par sa simplicité et recommandé dans le cadre des attaques par déni de service [129, 130, 131]. Il est cependant inefficace contre du trafic dont la source est falsifiée.

Le filtrage par liste noire repose sur un prérequis, à savoir, la capacité d'identifier et caractériser le trafic à rejeter. Les principaux types de format d'échange de données utilisés par des sondes réseau le permettent avec la granularité du flux IP (défini par les adresses IP source et destination). C'est le cas du format Extensible Event (EVE [1], Intrusion Detection Message Exchange Format (IDMEF) [144] ou encore Distributed Denial of Service Open Threat Signalling (DOTS) [2].

Les buts principaux du filtrage par liste noire est de maximiser le trafic d'attaque rejeté par le filtre, et en parallèle de minimiser les dégâts collatéraux. Dans un environnement réel, les listes noires sont stockées sur les équipements dans un espace limité, restreignant ainsi le nombre de règles de filtrage. Le nombre de règles pouvant être inférieur au nombre de flux IP malicieux, maximiser le trafic filtré nécessite d'agréger des flux IP malicieux.

Les dégâts sont induits par la définition des règles de filtrage, qui par leur granularité recouvre aussi du trafic légitime. La réduction des dégâts collatéraux requiert donc une connaissance du trafic légitime. Cependant aucune information, concernant le trafic légitime seulement, n'est disponible dans le réseau, celles-ci se rapportant essentiellement au regroupement du trafic légitime et malicieux.

[1] EVE v4.0.1, http://suricata.readthedocs.io/en/latest/output/eve/index.html

[2] https ://datatracker.ietf.org/wg/dots/about/

La visibilité d'un opérateur sur son réseau est dépendante de plusieurs facteurs : sa politique de surveillance, ses équipements. De la même manière, le niveau de détail concernant le trafic malicieux, contenu dans les alertes dépend des équipements permettant la détection de l'attaque. Nous définissons trois scénarios décrivant les différents niveaux de visibilité auquel un opérateur peut avoir accès. Ils sont exposés dans le Tableau A.1.

TABLE A.1 – Scénarios reflétant les différents niveaux de visibilité d'un opérateur sur son réseau

Scenario	**Description**	**Identification du trafic malicieux**	**Télémétries du trafic malicieux**	**Télémétries globales de la surveillance**
1	Prérequis minimum	Oui	Non	Non
2	Détection améliorée	Oui	Oui	Non
3	Entière visibilité	Oui	Oui	Oui

Le *prérequis minimum* décrit les informations minimales nécessaires à la génération de listes noires, à savoir, la caractérisation des flux IP de l'attaque, extraite des alertes de sécurité. La *détection améliorée* dépeint une situation ou l'opérateur à aussi accès aux télémétries se rapportant aux flux IP malicieux. La *visibilité entière* se rapporte à l'utilisation de télémétries sur le trafic global (légitime et malicieux) dans le réseau, et ce, dans le but de réduire les dégâts collatéraux.

A.3.1 Génération de listes noires appliquée à la couche IP

Cette section présente la génération de listes noires dans le but d'atteindre les objectifs cités précédemment : maximiser le trafic malicieux filtré et minimiser les dégâts collatéraux. La Figure A.9 schématise le déroulement de cette génération. Comme le nombre de règles de filtrage est limité, la capacité d'une liste noire à filtre le trafic dépend de la capacité d'agréger les flux IP malicieux lors de sa génération. Les agrégats sont formés en regroupant sous un même préfixe réseau les adresses IP sources des flux destinés à une même destination. Nous produisons dans un premier temps tous les agrégats possibles à partir de la liste des flux IP malicieux. Dans un deuxième temps, nous donnons un score à chaque agrégat. Ce score dépend des informations de télémétrie du trafic malicieux dans le cas du scénario 2 et 3, mais aussi des télémétries globales dans le scénario 3. Dans un troisième et dernier temps, nous ordonnons les agrégats et sélectionnons ceux avec le plus haut score, dans la limite du nombre de règles contraint par l'équipement.

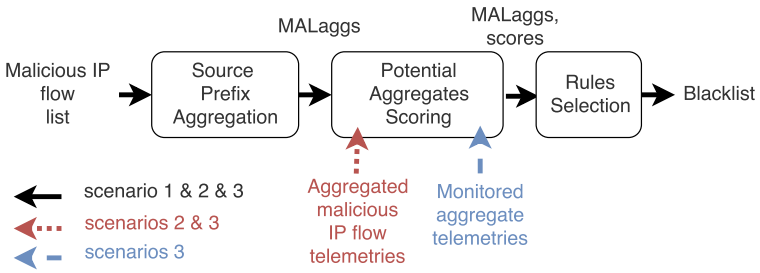


FIGURE A.9 – Processus de génération d'une liste noire

Nous définissons quatre simples fonctions permettant de calculer le score de chaque agrégat avec les informations disponibles. La fonction $score_{1.a}$, applicable dans le scénario 1, fait varier le score avec la taille du préfixe de l'agrégat source. Plus celui-ci est petit, et plus il va potentiellement contenir de flux IP malicieux.

La fonction $score_{1.b}$, elle aussi applicable dans le scénario 1, effectue le ratio entre le nombre de flux IP malicieux effectivement inclus dans l'agrégat source et sa taille. Il vise à favoriser l'ajout d'agrégat dont la concentration est plus importante.

La fonction $score_2$, applicable au scénario 2, effectue la somme des volumétries des flux IP malicieux inclus dans cet agrégat. L'impact d'un agrégat est donc lié à sa volumétrie, sachant que nous cherchons à atténuer la congestion dans le réseau lié à une saturation de la bande passante.

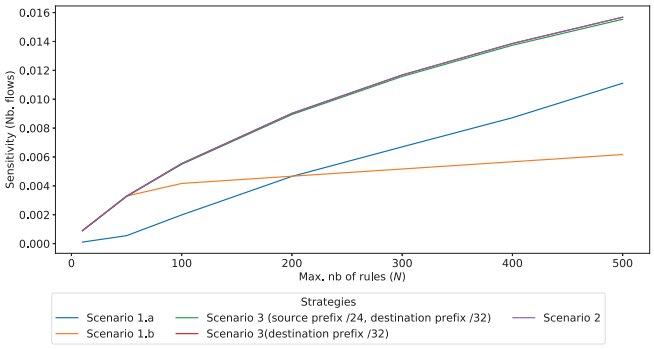
La fonction $score_3$, applicable au scénario 3, est un ratio entre la volumétrie du trafic malicieux et celle du trafic global inclue dans l'agrégat. Cette fonction privilégie les agrégats à fort volume malicieux, mais diminue leur impact lorsque le volume de trafic légitime augmente.

A.3.2 Évaluation du filtrage par liste noire

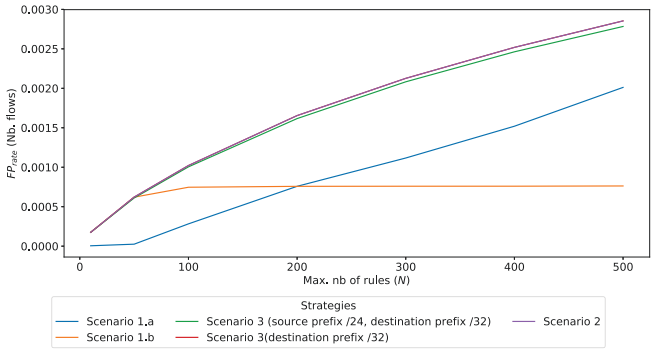
Nous évaluons l'efficacité du filtrage avec un trafic malicieux composé de 200 000 flux, chacun ayant un débit constant pendant toute la durée de l'attaque. L'efficacité est évaluée en termes de taux de vrais et faux positifs, notés respectivement TP_r et FP_r. Le taux de vrais positifs évalue la proportion de trafic malicieux filtré. Il correspond au ratio entre le nombre de flux IP malicieux filtré et le nombre total de flux malicieux. Le taux de faux positifs mesure la proportion des dégâts collatéraux. Il est égal au ratio entre le nombre de flux IP légitimes filtrés et le nombre total de flux IP légitimes.

Les Figures A.10a et A.10b montrent respectivement les taux de vrais et faux positifs en ordonnées résultant du filtrage du trafic d'attaque avec différentes limites en termes de nombre de règles dans une liste noire (axe des abscisses).

Nous voyons clairement que les fonctions de score du scénario 1 ont



(a) TP_r (nombre de flux)



(b) FP_r (nombre de flux)

FIGURE A.10 – Comparaison des fonctions de scores (agrégation minimale des adresses IP sources : /24)

des taux de vrais et faux positifs en deçà des autres fonctions de score. Ils filtrent moins de trafic malicieux, mais préservent plus de trafic légitime. La fonction $score_{1.a}$ n'est pas capable de sélectionner les agrégats malicieux les plus impactant. Ceci est dû au fait que beaucoup d'agrégats (190 000) obtiennent le même score. Il n'y a donc pas de méthode rationnelle pour sélectionner jusqu'à 500 agrégats à ajouter dans le filtre avec cette fonction.

La fonction $score_1.b$ semble voir sa croissance chuter à partir d'un nombre de règle supérieur à 50. Ceci est dû à la distribution du trafic, où les agrégats filtrés sont les plus denses en termes de volumétrie de trafic malicieux. Les agrégats avec un score inférieur contiennent moins de trafic malicieux.

La fonction $score_2$ coïncide avec la courbe de la fonction $score_{1.b}$ dans la Figure A.10a. Dans ce cas, l'utilisation des télémétries du trafic malicieux n'a aucune valeur ajoutée. À l'inverse, le taux de faux positif augmente plus rapidement avec la fonction $score_2$ comparé à la fonction $score_{1.b}$ pour un nombre maximal de règle par filtre supérieur à 50. En contrepartie, le taux de faux positifs augmente aussi plus rapidement (*cf.* Figure A.10b) entrainant une augmentation du nombre de dégâts collatéraux. Ceci est dû au fait que

la fonction $score_2$ n'essaye pas de réduire le nombre de dégâts collatéraux, à l'inverse de la fonction $score_1.b$.

La fonction $score_3$ a été évaluée avec deux configurations du système de surveillance réseau différentes. La première courbe en rouge a été réalisée avec des enregistrements du trafic relatifs à des adresses IP destinations. La seconde, en vert, a été réalisée avec une granularité plus fine des enregistrements du trafic, *i.e.* < `préfixe source /24, adresse IP destination` >. La courbe rouge coïncide avec la courbe relative à la fonction $score_2$ pour le taux faux positifs et le taux de vrais positifs. Ceci est dû au fait que très peu d'agrégats filtrés contiennent à la fois du trafic légitime et malicieux, de telle sorte, que l'introduction des volumétries provenant de la surveillance réseau ($score_3$) à grosse maille n'a que peu de valeur ajoutée. Une surveillance réseau avec une granularité plus fine (courbe verte) à des résultats plus visibles, quoique faible en termes de nombre de flux (*i.e.* une baisse de 0.01% du nombre de flux légitime filtrés). Cette différence est plus notoire lorsqu'elle est exprimée en termes de volume (0.2% du nombre d'octet total).

A.3.3 Conclusion

Les expérimentations ont permis de valider l'impact de la visibilité sur l'efficacité du filtrage, à la fois en termes de trafic malicieux filtré, mais aussi de dégâts collatéraux. L'hypothèse, selon laquelle avoir plus d'information sur l'état de son réseau permettrait d'être plus efficace dans le filtrage, n'est pas démontrée. En effet, par exemple, l'utilisation d'enregistrements à forte granularité provenant du système de surveillance réseau n'a pas eut pour résultat de diminuer les dégâts collatéraux. Une telle constatation est importante, car un tel prérequis en termes de surveillance réseau peut se révéler coûteux pour un opérateur qui ne verrait alors aucun retour sur investissement en termes de trafic légitime préservé.

Les expérimentations ont été effectuées avec du trafic malicieux pour lequel chaque flux d'attaque avait un débit constant. Dans la réalité, chaque flux varie au cours du temps. C'est le cas particulier des attaques par déni de service de type pulsante, c'est-à-dire que le trafic montre régulièrement des pics de volumétrie. L'efficacité des listes noires doit aussi être étudiée avec ce type de dynamique d'attaque. Nous estimons que de nouveaux paramètres d'études apparaitrons comme la période de rafraichissement des listes noires, afin de maintenir ces listes à jour ; ainsi que la période de collection des enregistrements du trafic, permettant une mise à jour des flux et de leur volumétrie à l'intérieur du réseau.

A.4 Conclusion

Dans cette thèse, nous avons proposé un mécanisme de partage de charge basé sur le standard Multiprotocol Label Switching (MPLS) destiné à isoler le trafic malicieux du trafic légitime. La séparation du trafic est effectuée à partir d'une signature de l'attaque qui est contenu dans un label MPLS, le Label de Mitigation. La signature a été générée en utilisant des informations de volumétrie concernant le trafic malicieux et légitime, mais aussi la bande passante disponible. Un routeur, responsable du partage de charge, transmet le trafic malicieux et légitime à travers respectivement un lien de mitigation et un lien prioritaire, en fonction du Label de Mitigation. Le trafic malicieux, ainsi isolé, peut recevoir un traitement plus restrictif que le trafic légitime lors de son acheminement à travers le réseau. Ce traitement différencié entre trafic malicieux et légitime est mis en place à partir des fonctions de transmission communément implémentées dans les équipements réseaux, comme défini dans l'extension MPLS Differentiated Services.

Nous avons montré que notre technique est efficace comparé à un mécanisme de partage de charge classique. Il est capable de préserver jusqu'à 50% de trafic légitime en plus lors d'une attaque saturant l'ensemble des liens. Cependant, si l'utilisation d'un label MPLS à des fins de partage de charge a été standardisé, notre mécanisme doit être implémenté dans les routeurs responsables du partage de charge.

Nous avons aussi étudié l'utilisation de listes noires contre les attaques par déni de service volumétriques. Les prérequis en termes de capacité de filtrage des équipements réseau ont été minimisés, de sorte que le filtrage ne s'applique qu'aux champs adresses IP source et destination. Ainsi notre étude s'applique à un très vaste panel de routeurs. L'efficacité des listes noires a été évaluée en termes de capacité à filtrer le trafic malicieux et à préserver le trafic légitime. À partir de la constatation que tous les opérateurs de réseau ne disposent pas de la même visibilité sur leur réseau, nous avons formalisé la notion de visibilité selon trois critères : la capacité d'identifier le trafic malicieux, d'obtenir des informations de volumétrie sur le trafic malicieux et sur l'ensemble du trafic dans le réseau. L'étude a montré notamment, qu'utiliser plus d'informations pour générer les listes noires n'impliquaient pas toujours une meilleure efficacité des filtres.

Les perspectives de travaux futurs s'orientent autour de deux axes : l'amélioration du mécanisme de partage de charge et l'utilisation des résultats de l'étude sur les listes noires pour aider à décider de la contre-mesure à mettre en place en cas d'attaque par déni de service.

Glossary

CoT a subset of traffic that define, for example, an application, protocol families, destniation or quaity of service requirements.. 23, 34

granularity reflects the amount of details, *e.g.* packets fields, a mechanism is able to act. 33

IPf a stream of packets, sharing the tuple < source IP, destination IP >. Defined this way, the flow includes only one direction of traffic and for example, a TCP connection will result in two IP flows.. 63, 102–104, 106, 108–110, 112–114, 116, 117

QoE or Quality of User Experience, depicts the perception of users regarding the quality of a service.. 10

Bibliography

[1] OVH. The DDoS that didn't break the camel's VAC, 2016.

[2] Michael E Whitman and Herbert J Mattord. *Principles of Information Security*. Course Technology Press, Boston, MA, United States, 4th edition, 2011.

[3] Over a Decade of DDoS. Technical report, Arbor Networks, 2012.

[4] Brian Krebs. KrebsOnSecurity Hit With Record DDoS, 2016.

[5] Darren Anstee, C.F Chui, and Paul Bowen. Wordwilde Infrastructure Security Report 2016. Technical report, Arbor Networks, 2016.

[6] Jelena Mirkovic and Peter L Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *Computer Communication Review*, 34(2):39–53, 2004.

[7] C Douligeris and A Mitrokotsa. DDoS attacks and defense mechanisms: a classification. In *Signal Processing and Information Technology, 2003. ISSPIT 2003. Proceedings of the 3rd IEEE International Symposium on*, pages 190–193, 2003.

[8] Abbass Asosheh Dr. and Naghmeh Ramezani. A Comprehensive Taxonomy of DDOS Attacks and Defense Mechanism Applying in a Smart Classification. *WSEAS Trans. on Comp.*, 7(4):281–290, 2008.

[9] Jelena Mirkovic, Alefiya Hussain, Sonia Fahmy, Peter L Reiher, and Roshan K Thomas. Accurately Measuring Denial of Service in Simulation and Testbed Experiments. *{IEEE} Trans. Dependable Sec. Comput.*, 6(2):81–95, 2009.

[10] Steve Mansfield-Devine. Anonymous: serious threat or mere annoyance? *Network Security*, 2011(1):4–10, jan 2011.

[11] Christian Rossow. Amplification Hell: Revisiting Network Protocols for DDoS Abuse. In *In Proceedings of the 2014 Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2014.

[12] Moti Geva. *Ensuring QoS During Bandwidth DDoS Attacks*. phdthesis, Bar-Ilan University, 2013.

[13] Nabil Hachem, Hervé Debar, and Joaquín García-Alfaro. HADEGA: A novel MPLS-based mitigation solution to handle network attacks. In *IPCCC*, pages 171–180. IEEE, 2012.

[14] M Li, J Li, and W Zhao. Simulation Study of Flood Attacking of DDOS. In *2008 International Conference on Internet Computing in Science and Engineering2008 International Conference on Internet Computing in Science and Engineering*, pages 286–293, jan 2008.

[15] Changwang Zhang, Jianping Yin, and Zhiping Cai. RSFB: a Resilient Stochastic Fair Blue algorithm against spoofing DDoS attacks. In *Communications and Information Technology, 2009. ISCIT 2009. 9th International Symposium on*, pages 1566–1567, sep 2009.

[16] Harkeerat Singh Bedi, Sankardas Roy, and Sajjan G Shiva. Mitigating congestion based DoS attacks with an enhanced {AQM} technique. *Computer Communications*, 56:60–73, 2015.

[17] C M Patel. URED: Upper threshold RED an efficient congestion control algorithm. In *Computing, Communications and Networking Technologies (ICCCNT),2013 Fourth International Conference on*, pages 1–5, 2013.

[18] P Owezarski. On the impact of DoS attacks on Internet traffic characteristics and QoS. In *Proceedings. 14th International Conference on Computer Communications and Networks, 2005. ICCCN 2005.*, pages 269–274, oct 2005.

[19] Philippe Owezarski and Nicolas Larrieu. Internet Traffic Characterization – An Analysis of Traffic Oscillations. In Zoubir Mammeri and Pascal Lorenz, editors, *High Speed Networks and Multimedia Communications: 7th IEEE International Conference, HSNMC 2004, Toulouse, France, June 30 - July 2, 2004. Proceedings*, chapter Internet m, pages 96–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[20] Srinivasan Keshav. *Congestion Control in Computer Networks*. PhD thesis, EECS Department, University of California, Berkeley, sep 1991.

[21] G.1010 End-user multimedia QoS categories. Technical report, ITU, 2001.

[22] J Mirkovic, A Hussain, B Wilson, S Fahmy, Roshan Thomas, Wei-Min Yao, and Stephen Schwab. Towards User-Centric Metrics for Denial-Of-Service Measurement. *Proceedings of the 2007 Workshop on Experimental Computer Science*, (June):13–14, 2007.

[23] Ari Juels and John G Brainard. Client Puzzles: {A} Cryptographic Countermeasure Against Connection Depletion Attacks. In *Proceedings of the Network and Distributed System Security Symposium, {NDSS} 1999, San Diego, California, {USA}*, 1999.

[24] XiaoFeng Wang and Michael K Reiter. Mitigating bandwidth-exhaustion attacks using congestion puzzles. In *Proceedings of the 11th {ACM} Conference on Computer and Communications Security, {CCS} 2004, Washington, DC, USA, October 25-29, 2004*, pages 257–267, 2004.

[25] Soon Hin Khor and Akihiro Nakao. DaaS: DDoS Mitigation-as-a-Service. In *2011 IEEE/IPSJ International Symposium on Applications and the Internet*, pages 160–171. IEEE, jul 2011.

[26] Christoph L Schuba, Ivan Krsul, Markus G Kuhn, Eugene H Spafford, Aurobindo Sundaram, and Diego Zamboni. Analysis of a Denial of Service Attack on {TCP}. In *1997 {IEEE} Symposium on Security and Privacy, May 4-7, 1997, Oakland, CA, {USA}*, pages 208–223, 1997.

[27] Tony Miu, Albert Hui, Wai Leng Lee, and Alan Chung. Bypassing DDoS Mitigation. In *PacSec2013*, Tokyo, 2013.

[28] Suphannee Sivakorn, Iasonas Polakis, and Angelos D. Keromytis. I am robot: (deep) learning to break semantic image captchas. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 388–403. IEEE, 2016.

[29] Abraham Yaar, Adrian Perrig, and Dawn Xiaodong Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy*, pages 130–. IEEE Computer Society, 2004.

[30] Xiaowei Yang, David Wetherall, and Thomas E Anderson. TVA: A DoS-limiting network architecture. In Roch Guérin, Ramesh Govindan, and Greg Minshall, editors, *SIGCOMM*, volume 16, pages 241–252. ACM, dec 2005.

[31] Michael Welzl. *Network Congestion Control - Managing Internet Traffic*. Wiley, 2005.

[32] J. Nagle. Congestion Control in IP/TCP Internetworks. RFC 896, January 1984.

[33] Cisco Systems. Remotely triggered black hole filtering - destination based and source based. Technical report, Cisco Systems, 2005.

[34] Ioannis Vordos. *Mitigating Distributed Denial of Service Attacks with Multi-Protocol Label Switching-Traffic Engineering (MPLS-TE)*. PhD thesis, Naval Postgraduate School, 2009.

[35] Cisco. Access Control Lists: Overview and Guidelines. Technical report.

[36] Fabio Soldo, Katerina Argyraki, and Athina Markopoulou. Optimal Source-Based Filtering of Malicious Traffic. *IEEE/ACM Transactions on Networking*, 20(2):381–395, apr 2012.

[37] Understanding ACL on catalyst 6500 series switches. Technical report, Cisco.

[38] Mingwei Xu, Shu Yang, Dan Wang, Fuliang Li, and Jianping Wu. Source address filtering for large scale networks. *Computer Communications*, 2014.

[39] Gary Pack, Jaeyoung Yoon, Eli Collins, and Cristian Estan. On Filtering of DDoS Attacks Based on Source Address Prefixes. In *2006 Securecomm and Workshops*, pages 1–12. IEEE, aug 2006.

[40] L. Maccari, R. Fantacci, P. Neira, and R. M. Gasca. Mesh Network Firewalling with Bloom Filters. In *2007 IEEE International Conference on Communications*, pages 1546–1551. IEEE, jun 2007.

[41] Burton H Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, 1970.

[42] Markus Goldstein, Christoph Lampert, Matthias Reif, Armin Stahl, and Thomas Breuel. Bayes Optimal DDoS Mitigation by Adaptive History-Based IP Filtering. In *Seventh International Conference on Networking (icn 2008)*, pages 174–179. IEEE, apr 2008.

[43] Aapo Kalliola, Tuomas Aura, and Sanja Šćepanović. Denial-of-Service Mitigation for Internet Services. pages 213–228. Springer, Cham, oct 2014.

[44] Balachander Krishnamurthy, Jia Wang, Balachander Krishnamurthy, and Jia Wang. On network-aware clustering of Web clients. *ACM SIGCOMM Computer Communication Review*, 30(4):97–110, oct 2000.

[45] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. Finding Hierarchical Heavy Hitters in Data Streams. *VLDB '03 - Proceedings of the twenty-ninth VLDB conference*, 29:464–475, 2003.

[46] Simon Hauger, Thomas Wild, Arthur Mutter, Andreas Kirstaedter, Kimon Karras, Rainer Ohlendorf, Frank Feller, and Joachim Scharf. Packet Processing at 100 Gbps and Beyond - Challenges and Perspectives. *Photonic Networks, 2009 ITG Symposium on*, pages 1–10, 2009.

[47] Robin Sommer, Vern Paxson, and Nicholas Weaver. An architecture for exploiting multi-core processors to parallelize network intrusion prevention. *Concurrency and Computation: Practice and Experience*, 21(10):1255–1279, 2009.

[48] Nigel Jacob and Carla Brodley. Offloading {IDS} Computation to the {GPU}. In *Proceedings of the 22nd Annual Computer Security Applications Conference on Annual Computer Security Applications Conference (ACSAC '06)*, pages 371–380, Washington, DC, USA, 2006. IEEE Computer Society.

[49] Nen-Fu Huang, Hsien-Wei Hung, Sheng-Hung Lai, Yen-Ming Chu, and Wen-Yen Tsai. A gpu-based multiple-pattern matching algorithm for network intrusion detection systems. In *AINA Workshops*, pages 62–67. IEEE Computer Society, 2008.

[50] Giorgos Vasiliadis, Spyros Antonatos, Michalis Polychronakis, Evangelos P. Markatos, and Sotiris Ioannidis. Gnort: High performance network intrusion detection using graphics processors. In Richard Lippmann, Engin Kirda, and Ari Trachtenberg, editors, *RAID*, volume 5230 of *Lecture Notes in Computer Science*, pages 116–134. Springer, 2008.

[51] Randy Smith, Neelam Goyal, Justin Ormont, Karthikeyan Sankaralingam, and Cristian Estan. Evaluating gpus for network packet signature matching. In *ISPASS*, pages 175–184. IEEE Computer Society, 2009.

[52] Tamás Tóthfalusi and Péter Orosz. Line-rate packet processing in hardware: the evolution towards 400 Gbit/s. In *Proceedings of the 9th International Conference on Applied Informatics, Volume 1*, volume 1, pages 259–268, 2015.

[53] Martino Trevisan, Alessandro Finamore, Marco Mellia, Maurizio Munafò, Dario Rossi, and Politecnico Di Torino. DPDKStat: 40Gbps Statistical Traffic Analysis with Off-the-Shelf Hardware. Technical report, 2016.

[54] Scott Gerlach and Don LeBert. Mitigating denial of service attacks.

[55] Nirwan Ansari and Zhiqiang Gao. Behavior-based Traffic Differentiation (BTD) for Defending against Dstributed Denial of Service (DDoS) Attacks, 2012.

[56] Yuta Tokusashi, Yohei Kuga, Ryo Nakamura, Hajime Tazaki, and Hiroki Matsutani. mitiKV: An Inline Mitigator for DDoS Flooding Attacks. *Internet Conference 2016*, 2016.

[57] Vinayaka Jyothi, Sateesh K. Addepalli, and Ramesh Karri. Deep Packet Field Extraction Engine (DPFEE): A pre-processor for network intrusion detection and denial-of-service detection systems. In *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pages 266–272. IEEE, oct 2015.

[58] Pavel Benacek, Viktor Pus, and Hana Kubatova. P4-to-VHDL: Automatic Generation of 100 Gbps Packet Parsers. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 148–155, may 2016.

[59] The P4 Language Consortium. The P4 Language Specification 1.0.4. Technical report, 2017.

[60] H Wang, C Jin, and K G Shin. Defense Against Spoofed IP Traffic Using Hop-Count Filtering. *IEEE/ACM Transactions on Networking*, 15(1):40–53, feb 2007.

[61] Scott Fowler, Sherali Zeadally, and Naveen Chilamkurti. Impact of denial of service solutions on network quality of service. *Security and Communication Networks*, 4(10):1089–1103, 2011.

[62] S Floyd and V Jacobson. Link-sharing and resource management models for packet networks. *{IEEE/ACM} Transactions on Networking*, 3(4):365–386, 1995.

[63] a. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM Computer Communication Review*, volume 19, pages 1–12, 1989.

[64] P E McKenney. Stochastic fairness queueing. In *INFOCOM '90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE*, pages 733–740 vol.2, jun 1990.

[65] Ion Stoica, Scott Shenker, and Hui Zhang. Core-stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High

Speed Networks. In *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '98, pages 118–130, New York, NY, USA, 1998. ACM.

[66] M Shreedhar and George Varghese. Efficient fair queueing using deficit round-robin. *{IEEE/ACM} Trans. Netw.*, 4(3):375–385, 1996.

[67] Sivasubramaniam Nandhini and Senniappan Palaniammal. Enhanced core stateless fair queuing with multiple queue priority scheduler. *Int. Arab J. Inf. Technol.*, 11(2):159–167, 2014.

[68] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *{IEEE/ACM} Trans. Netw.*, 1(4):397–413, 1993.

[69] Sally Floyd, Ramakrishna Gummadi, Scott Shenker, and Others. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management, 2001.

[70] James Aweya, Michel Ouellette, and Delfin Y Montuno. A control theoretic approach to active queue management. *Computer Networks*, 36(2-3):203–235, jul 2001.

[71] Dong Lin and Robert Morris. Dynamics of Random Early Detection. In *Proceedings of the {ACM} {SIGCOMM} 1997 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, September 14-18, 1997, Cannes, France.*, pages 127–137, 1997.

[72] T J Ott, T V Lakshman, and L H Wong. SRED: stabilized RED. In *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1346–1355 vol.3, mar 1999.

[73] R Mahajan, S Floyd, and D Wetherall. Controlling high-bandwidth flows at the congested router. In *Network Protocols, 2001. Ninth International Conference on*, pages 192–201, nov 2001.

[74] Shan Chen, Zhen Zhou, and Brahim Bensaou. Stochastic {RED} and Its Applications. In *Proceedings of {IEEE} International Conference on Communications, {ICC} 2007, Glasgow, Scotland, 24-28 June 2007*, pages 6362–6367, 2007.

[75] Rong Pan, B Prabhakar, and K Psounis. CHOKe - a stateless active queue management scheme for approximating fair bandwidth allocation. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the*

IEEE Computer and Communications Societies. Proceedings. IEEE, volume 2, pages 942–951 vol.2, 2000.

[76] P Chhabra, S Chuig, A Goel, A John, A Kumar, H Saran, and R Shorey. XCHOKe: malicious source control for congestion avoidance at Internet gateways. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 186–187, nov 2002.

[77] V V Govindaswamy, G Zaruba, and G Balasekaran. RECHOKe: A Scheme for Detection, Control and Punishment of Malicious Flows in IP Networks. In *IEEE GLOBECOM 2007 - IEEE Global Telecommunications Conference*, pages 16–21, nov 2007.

[78] Tetsuji Yamagushi. *A queue management algorithm for fair bandwidth allocation by controlling bursty traffic with stateless information and its performance evaluation.* PhD thesis, Kyoto University, 2006.

[79] Wu-chang Feng, K G Shin, D D Kandlur, and D Saha. The BLUE active queue management algorithms. *IEEE/ACM Transactions on Networking*, 10(4):513–528, aug 2002.

[80] Wu-Chang Feng, D D Kandlur, D Saha, and K G Shin. Stochastic fair blue: a queue management algorithm for enforcing fairness. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1520–1529 vol.3, 2001.

[81] Erika P Alvarez-Flores, Juan J Ramos-Munoz, Pablo Ameigeiras, and Juan M Lopez-Soler. Selective packet dropping for VoIP and TCP flows. *Telecommunication Systems*, 46(1):1–16, 2011.

[82] Wu-chang Feng, Dilip D Kandlur, Debanjan Saha, and Kang G Shin. BLUE: A New Class of Active Queue Management Algorithms. *U. Michigan CSE-TR-387-99*, 1999.

[83] Cisco. Comparing Traffic Policing and Traffic Shaping for Bandwidth Limiting, 2008.

[84] Cisco IOS Quality of Service Solutions Configuration Guide, Release 12.2 - Policing and Shaping Overview.

[85] Ratul Mahajan, Steven M. Bellovin, Sally Floyd, John Ioannidis, Vern Paxson, and Scott Shenker. Controlling high bandwidth aggregates in the network. *Computer Communication Review*, 32(3):62–73, 2002.

[86] John Ioannidis and Steven M. Bellovin. Implementing pushback: Router-based defense against ddos attacks. In *NDSS*. The Internet Society, 2002.

[87] Xin Liu, Xiaowei Yang, and Yanbin Lu. To filter or to authorize: network-layer dos defense against multimillion-node botnets. In Victor Bahl, David Wetherall, Stefan Savage, and Ion Stoica, editors, *SIGCOMM*, pages 195–206. ACM, 2008.

[88] Yehoshua Gev, Moti Geva, and Amir Herzberg. Backward traffic throttling to mitigate bandwidth floods. In *GLOBECOM - IEEE Global Telecommunications Conference*, pages 904–910, 2012.

[89] Job Snijders. DDoS Damage Control Cheap & effective. In *RIPE 68*, Warvsaw, 2014.

[90] Sotiris Ioannidis, Angelos D. Keromytis, Steven M. Bellovin, and Jonathan M. Smith. Implementing a distributed firewall. In Dimitris Gritzalis, Sushil Jajodia, and Pierangela Samarati, editors, *ACM Conference on Computer and Communications Security*, pages 190–199. ACM, 2000.

[91] Sharad Agarwal, Travis Dawson, and C Tryfonas. DDoS mitigation via regional cleaning centers. *Sprint ATL Research Report*, (January):1–11, 2003.

[92] C J Fung and B McCormick. VGuard: A distributed denial of service attack mitigation method using network function virtualization. In *Network and Service Management (CNSM), 2015 11th International Conference on*, pages 64–70, nov 2015.

[93] J Mirkovic, G Prier, and P Reiher. Attacking DDoS at the source. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference on*, pages 312–321, nov 2002.

[94] Haining Wang and Kang G. Shin. Transport-aware IP routers: A built-in protection mechanism to counter DDoS attacks. *IEEE Trans. Parallel Distrib. Syst.*, 14(9):873–884, 2003.

[95] Eric Y. K. Chan, H. W. Chan, K. M. Chan, Vivien P. S. Chan, Samuel T. Chanson, Matthew M. H. Cheung, C. F. Chong, K. P. Chow, Albert K. T. Hui, Lucas Chi Kwong Hui, Luke C. K. Lam, W. C. Lau, Kevin K. H. Pun, Anthony Y. F. Tsang, Wai Wan Tsang, Sam C. W. Tso, Dit-Yan Yeung, and Kwun Yin Yu. Idr: An intrusion detection router for defending against distributed denial-of-service (ddos) attacks. In *ISPAN*, pages 581–586. IEEE Computer Society, 2004.

[96] F. Baker and P. Savola. Ingress Filtering for Multihomed Networks. RFC 3704 (Best Current Practice), March 2004.

[97] S. Hanks, T. Li, D. Farinacci, and P. Traina. Generic Routing Encapsulation (GRE). RFC 1701 (Informational), October 1994.

[98] J. Lau, M. Townsley, and I. Goyret. Layer Two Tunneling Protocol - Version 3 (L2TPv3). RFC 3931 (Proposed Standard), March 2005. Updated by RFC 5641.

[99] Philip Smith, Rob Evans, and Mike Hughes. RIPE Routing Working Group Recommendations on Route Aggregation, 2006.

[100] Gero Schollmeier, Joachim Charzinski, and Andreas Kirstadter. Improving the Resilience in IP Networks. In *High Performance Switching and routing, 2003, HPSR. Workshop on*, pages 91–96, jun 2003.

[101] Michael Menth, Andreas Reifert, and Jens Milbrandt. Self-protecting multipaths - A simple and resource-effcient protection switching mechanism for MPLS networks. In *NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communication, Third International IFIP-TC6 Networking Conference, Athens, Greece, May 9-14, 2004, Proceedings*, pages 526–537, 2004.

[102] Shree Murthy and J. J. Garcia-Luna-Aceves. Congestion-oriented shortest multipath routing. In *Proceedings IEEE INFOCOM '96, The Conference on Computer Communications, Fifteenth Annual Joint Conference of the IEEE Computer and Communications Societies, Networking the Next Generation, San Francisco, CA, USA, March 24-28, 1996*, pages 1028–1036, 1996.

[103] Deliverable D1.4: Threat Data Final Report. Technical report, NECOMA, 2016.

[104] J. Moy. OSPF Version 2. RFC 2328 (INTERNET STANDARD), April 1998. Updated by RFCs 5709, 6549, 6845, 6860, 7474.

[105] Nayyar Almas Kazmi, Arie M. C. A. Koster, and Jürgen Branke. Formulations and algorithms for the multi-path selection problem in network routing. In *ICUMT*, pages 738–744. IEEE, 2012.

[106] Junjie Zhang, Kang Xi, Liren Zhang, and H. Jonathon Chao. Optimizing network performance using weighted multipath routing. In *21st International Conference on Computer Communications and Networks (ICCCN), 2012*, pages 1–7, July 2012.

[107] Manolis Katevenis, Stefanos Sidiropoulos, and Costas Courcoubetis. Weighted round-robin cell multiplexing in a general-purpose ATM switch chip. *IEEE Journal on Selected Areas in Communications*, 9(8):1265–1279, 1991.

[108] Zhiruo Cao, Zheng Wang, and Ellen W. Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM*, pages 332–341. IEEE, 2000.

[109] International Organization for Standardization. Information Processing Systems = Data Communication High = Level Data Link Control Procedure = Frame Structure. International standard; ISO 3309 ISO 3309: 1984, International Organization for Standardization, 1984.

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

[111] Richard Steenbergen. MPLS for Dummies, 2010.

[112] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702 (Informational), September 1999.

[113] F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services. RFC 3270 (Proposed Standard), May 2002. Updated by RFC 5462.

[114] Hachem Hachem, Nabil. *MPLS-based mitigation technique to handle cyber attacks*. Theses, Institut National des Télécommunications, July 2014.

[115] David G Andersen, Hari Balakrishnan, M Frans Kaashoek, and Robert Tappan Morris. Resilient Overlay Networks. In Keith Marzullo and M Satyanarayanan, editors, *Proceedings of the 18th {ACM} Symposium on Operating System Principles, {SOSP} 2001, Chateau Lake Louise, Banff, Alberta, Canada, October 21-24, 2001*, pages 131–145. ACM, 2001.

[116] Angelos D Keromytis, Vishal Misra, and Dan Rubenstein. {SOS:} secure overlay services. In *Proceedings of the {ACM} {SIGCOMM} 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication, August 19-23, 2002, Pittsburgh, PA, {USA}*, pages 61–72, 2002.

[117] Adam Greenhalgh, Mark Handley, and Felipe Huici. Using routing and tunneling to combat DoS attacks. In Dina Katabi and Balachander Krishnamurthy, editors, *SRUTI*. USENIX Association, 2005.

[118] T. Karagiannis, M. Molle, and M. Faloutsos. Long-range dependence ten years of Internet traffic modeling. *IEEE Internet Computing*, 8(5):57–64, sep 2004.

[119] Andrew S Tanenbaum and David J Wetherall. *Computer Networks*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition, 2010.

[120] J. Gozdecki, A. Jajszczyk, and R. Stankiewicz. Quality of service terminology in IP networks. *IEEE Communications Magazine*, 41(3):153–159, mar 2003.

[121] Christian Grimm and Georg Schlüchtermann. *IP Traffic Theory and Performance*. Signals and Communication Technology. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[122] William B. Norton. What are the historical transit pricing trends?

[123] Jacobus Van Der Merwe, Subhabrata Sen, and Charles Kalmanek. Streaming Video Traffic: Characterization and Network Impact. In *In Proc. of International Web Content Caching and Distribution Workshop*, 2002.

[124] AWS Best Practices for DDoS Resiliency. Technical Report June, Amazon Web Services, 2016.

[125] Massimo Ficco and Massimiliano Rak. Economic Denial of Sustainability Mitigation in Cloud Computing. In Cecilia Rossignoli, Mauro Gatti, and Rocco Agrifoglio, editors, *Organizational Innovation and Change*, pages 229–238, Cham, 2016. Springer International Publishing.

[126] M Naresh Kumar, P Sujatha, V Kalva, R Nagori, A K Katukojwala, and M Kumar. Mitigating Economic Denial of Sustainability (EDoS) in Cloud Computing Using In-cloud Scrubber Service. In *2012 Fourth International Conference on Computational Intelligence and Communication Networks*, pages 535–539, nov 2012.

[127] Eddy M.(Verizon Federal Network Systems) Wesley. Defenses Against TCP SYN Flooding Attacks. *The Internet Protocol Journal*, 9(4), 2006.

[128] Masoud Bekravi, Shahram Jamali, and Gholam Shaker. Defense against SYN-Flood Denial of Service Attacks Based on Learning Automata. *CoRR*, abs/1208.5, 2012.

[129] Comprendre et anticiper les attaques DDoS. Technical report, ANSSI, 2015.

[130] Que faire en cas d'attaque par déni de service (DDoS) ?, 2018.

[131] DDos Quick Guide. Technical report, National Cybersecurity and Communications Integration Center, 2014.

[132] Cisco. Configuring a Load-Balancing Scheme. In *IP Switching Cisco Express Forwarding Configuration Guide, Cisco IOS Release 15M&T*, pages 57—-72. 2018.

[133] Anvitha Prabhu, Shashank Singh, and Shridhar Dhodapkar. CEF polarization. Technical report, Cisco, 2013.

[134] Jeff Wheeler and Job Snijders. Understanding MPLS Hashing. In *NANOG 57*, Orlando, 2013.

[135] Job Snijders. What's up with poor performance towards MAC addresses starting with 4 or 6? 2013.

[136] S. Bryant, C. Filsfils, U. Drafz, V. Kompella, J. Regan, and S. Amante. Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network. RFC 6391 (Proposed Standard), November 2011. Updated by RFC 7274.

[137] K. Kompella, J. Drake, S. Amante, W. Henderickx, and L. Yong. The Use of Entropy Labels in MPLS Forwarding. RFC 6790 (Proposed Standard), November 2012. Updated by RFCs 7274, 7447.

[138] K. Kompella, L. Andersson, and A. Farrel. Allocating and Retiring Special-Purpose MPLS Labels. RFC 7274 (Proposed Standard), June 2014.

[139] Rüdiger Martin, Michael Menth, and Michael Hemmkeppler. Accuracy and dynamics of hash-based load balancing algorithms for multipath internet routing. In *BROADNETS*. IEEE, 2006.

[140] B. Matthews and P. Agarwal. Hash-based load balancing with per-hop seeding, September 13 2012. US Patent App. 13/418,283.

[141] A. Roitshtein and T. Mizrahi. Load balancing hash computation for network switches, June 17 2014. US Patent 8,756,424.

[142] V. Shah, N.S.S.N. Rao, A. Agrawal, S. Sarkar, K. Subramanian, and H. Shukla. System and method for balancing TCP/IP/workload of multi-processor system based on hash buckets, March 28 2006. US Patent 7,020,713.

[143] Juniper Networks. Inc. No Title. In *Junos ® OS MPLS Applications Feature Guide*, chapter Chapter 12. 2018.

[144] H. Debar, D. Curry, and B. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765 (Experimental), March 2007.

[145] Nik Teague. Open threat signaling using rpc api over https and ipfix. Internet-Draft draft-teague-open-threat-signaling-01, IETF Secretariat, July 2015. http://www.ietf.org/internet-drafts/draft-teague-open-threat-signaling-01.txt.

[146] I O S Cisco. NetFlow, 2008.

[147] Traffic Monitoring using sFlow, 2003.

[148] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek. Architecture for IP Flow Information Export. RFC 5470 (Informational), March 2009. Updated by RFC 6183.

[149] Shahabeddin Geravand and Mahmood Ahmadi. Bloom filter applications in network security: A state-of-the-art survey. *Computer Networks*, 57(18):4047–4064, 2013.

[150] Li Fan, Pei Cao, Jussara M Almeida, and Andrei Z Broder. Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *SIGCOMM*, pages 254–265, 1998.

[151] M. Behringer. Analysis of the Security of BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4381 (Informational), February 2006.

[152] Romain Fontugne, Pierre Borgnat, Patrice Abry, and Kensuke Fukuda. MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In Jaudelice Cavalcante de Oliveira, Maximilian Ott, Timothy G Griffin, and Muriel Médard, editors, *CoNEXT*, page 8. ACM, 2010.

[153] A Whitaker and D Wetherall. Forwarding without loops in Icarus. In *Proc. of OPENARCH*, 2002.

[154] Marek Majkowski. IP spoofing. In *Packet Hacking Village*, 2017.

[155] Anna-Senpai. Mirai Source Code.

[156] R. Gerhards. The Syslog Protocol. RFC 5424 (Proposed Standard), March 2009.

[157] Tirumaleswar Reddy, Mohamed Boucadair, Prashanth Patil, Andrew Mortensen, and Nik Teague. Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal Channel Specification. Internet-Draft draft-ietf-dots-signal-channel-19, Internet Engineering Task Force, April 2018. Work in Progress.

[158] Tirumaleswar Reddy, Mohamed Boucadair, Kaname Nishizuka, Liang Xia, Prashanth Patil, Andrew Mortensen, and Nik Teague. Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel Specification. Internet-Draft draft-ietf-dots-data-channel-16, Internet Engineering Task Force, May 2018. Work in Progress.

[159] Mahesh Jethanandani, Lisa Huang, Sonal Agarwal, and Dana Blair. Network Access Control List (ACL) YANG Data Model. Internet-Draft draft-ietf-netmod-acl-model-19, Internet Engineering Task Force, April 2018. Work in Progress.

[160] M. Bjorklund. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020 (Proposed Standard), October 2010.

[161] Cisco. Configuring NetFlow Aggregation Caches. In *NetFlow Configuration Guide, Cisco IOS XE Release 3S*, pages 1–40. 2018.

[162] Benoit Claise. NetFlow Services. In *RIPE 44*, 2001.

[163] Cisco. NetFlow Hardware Support. In *Cisco IOS Software Configuration Guide,*, chapter 49. 2016.

[164] Fabio Soldo, Anh Le, and Athina Markopoulou. Predictive Blacklisting as an Implicit Recommendation System. aug 2009.

[165] Aapo Kalliola, Kiryong Lee, Heejo Lee, and Tuomas Aura. Flooding DDoS mitigation and traffic management with software defined networking. In *Cloud Networking (CloudNet), 2015 IEEE 4th International Conference on*, pages 248–254. IEEE, oct 2015.

[166] M. Collins and M.K. Reiter. An empirical analysis of target-resident dos filters. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 103–114. IEEE, 2004.

[167] Devin Bayer. Visibility of Prefix Lengths in IPv4 and IPv6, 2010.

[168] Daniela Brauckhoff, Bernhard Tellenbach, Arno Wagner, Martin May, and Anukool Lakhina. Impact of packet sampling on anomaly detection metrics. In Jussara M. Almeida, Virgílio A. F. Almeida, and Paul Barford, editors, *Internet Measurement Conference*, pages 159–164. ACM, 2006.

[169] Julien Freudiger, Emiliano De Cristofaro, and Alex Brito. Controlled Data Sharing for Collaborative Predictive Blacklisting. feb 2015.

[170] José Luis García-Dorado, Felipe Mata, Javier Ramos, Pedro M. Santiago del Río, Victor Moreno, and Javier Aracil. High-performance network traffic processing systems using commodity hardware. In *Data Traffic Monitoring and Analysis*, pages 3–27. 2013.

[171] Pavel Benácek, Viktor Pus, Jan Korenek, and Michal Kekely. Line rate programmable packet processing in 100gb networks. In Marco D. Santambrogio, Diana Göhringer, Dirk Stroobandt, Nele Mentens, and Jari Nurmi, editors, *27th International Conference on Field Programmable Logic and Applications, FPL 2017, Ghent, Belgium, September 4-8, 2017*, page 1. IEEE, 2017.