NATIONAL TECHNICAL UNIVERSITY OF ATHENS
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
DIVISION OF COMMUNICATION, ELECTRONIC AND INFORMATION ENGINEERING
NETWORK MANAGEMENT AND OPTIMAL DESIGN LABORATORY

Intelligent Services for Detection and Mitigation of Distributed Denial-of-Service Attacks in Programmable Network Environments

Doctoral Dissertation

of

Marinos I. Dimolianis

Advisory Committee: Vasilis Maglaris, Emeritus Professor, NTUA

Athens, April 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΔΙΑΧΕΙΡΙΣΗΣ ΚΑΙ ΒΕΛΤΙΣΤΟΥ ΣΧΕΔΙΑΣΜΟΥ ΔΙΚΤΥΩΝ ΤΗΛΕΜΑΤΙΚΗΣ

Ευφυείς Μηχανισμοί Ανίχνευσης και Αντιμετώπισης Κατανεμημένων Επιθέσεων Άρνησης Παροχής Υπηρεσιών σε Προγραμματιζόμενες Δικτυακές Υποδομές

Διδακτορική Διατριβή

του

Μαρίνου Ι. Δημολιάνη

Συμβουλευτική Επιτροπή: Βασίλειος Μάγκλαρης, Καθηγητής ΕΜΠ (Επιβλέπων)

Ευστάθιος Συκάς, Καθηγητής ΕΜΠ

Νεκτάριος Κοζύρης, Καθηγητής ΕΜΠ

……..	……..	……..
Ευστάθιος Συκάς Καθηγητής, ΕΜΠ	Νεκτάριος Κοζύρης Καθηγητής, ΕΜΠ	Συμεών Παπαβασιλείου Καθηγητής, ΕΜΠ
……..	……..	……..
Γεώργιος Στάμου Καθηγητής, ΕΜΠ	John Baras Professor, UMD	Χρυσούλα Παπαγιάννη Assistant Professor, UvA

……..

Δημήτριος Καλογεράς
Ερευνητής ‘Α, ΕΠΙΣΕΥ

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 19η Απριλίου 2022.

Αθήνα, Απρίλιος 2022

……

Μαρίνος Ι. Δημολιάνης
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Abstract

In this dissertation, we leverage on capabilities offered by the Network Softwarization paradigm and combine them with advanced data analysis techniques, i.e. Machine Learning (ML), towards the development of an integrated protection framework against cyberattacks. We focus on Distributed-Denial of Service (DDoS) attacks and implement mechanisms for efficient network data collection, fast and reliable anomaly detection and effective mitigation.

Initially, we design a DDoS detection mechanism entirely offloaded in the data plane using the P4 language. Through traffic features computed and evaluated in-network, DDoS attacks victims are identified rapidly within short timeframes. Detection in the data plane is one step ahead of control plane mechanisms that stall real-time detection and mitigation of network attacks.

Detecting the victim of network attacks is only the first step towards mitigating them and is followed by traffic classification procedures. Thus, in this dissertation we introduce a novel signature-based classification and mitigation schema based on softwarized data planes, i.e. eXpress Data Path (XDP). Supervised Learning algorithms (Random Forests, Multilayer-Perceptrons), applied to packet features (signatures), segregate malicious from benign packets. The employed features are pre-selected through an automated process that eliminates inconsequential features. To expedite mitigation performance and ease filtering rules management, source IP-agnostic rules tailored to the attack traffic are generated. This is achieved via a multi-objective optimization problem formulation that reduces filtering rules number with minimal effect on benign traffic. The proposed signature-based mechanism is evaluated in two broad categories of DDoS attacks, protocol (i.e. SYN Flood) and volumetric (i.e. DNS Amplification). Based on experimental evaluations, our innovative approach outperforms the state-of-the-art flow-based protection mechanisms by (i) detecting attacks in shorter time-windows, (ii) optimizing the number and type of filtering rules, and (iii) achieving increased packet filtering performance.

Finally, in this dissertation, we extend our signature-based schema to collaborative network environments. Collaborative DDoS detection relies on Federated Learning techniques that enable for cooperative and privacy-aware learning. Collaborative DDoS

mitigation is implemented in programmable XDP-based middleboxes featuring a scalable, cost-effective protection as-a-service mechanism. By contrast to traditional protection schemes, we allow data exchange amongst disjoint network domains with respect to data privacy legislations; moreover, we offer a flexible yet efficient firewall solution offloaded in Commercial-off-the-Shelf hardware.

Our integrated protection framework is deployed in programmable network hardware and evaluated using production network data from diverse and heterogeneous network environments, featuring fully realistic experimentation.

Keywords:

DDoS attacks, Anomaly Detection, Attack Mitigation, Software-Defined Networking (SDN), Data Plane Programmability, P4, eXpress Data Path (XDP), Supervised Learning, Federated Learning

Περίληψη

Η παρούσα διδακτορική διατριβή ερευνά τεχνολογίες αιχμής των σύγχρονων δικτύων υπολογιστών με έμφαση στις δυνατότητες προγραμματισμού τους και παράλληλα εξετάζει μεθοδολογίες και αλγορίθμους ευφυούς ανάλυσης δικτυακών δεδομένων. Συνδυάζοντας αυτές τις δύο πτυχές έχει ως στόχο την δημιουργία ενός ολοκληρωμένου μηχανισμού προστασίας ενάντια σε κυβερνοεπιθέσεις. Συγκεκριμένα, ασχολείται με τις κατανεμημένες επιθέσεις άρνησης παροχής υπηρεσιών και μελετά μεθόδους αποδοτικής συλλογής δεδομένων, τεχνικές άμεσης και αξιόπιστης ανίχνευσης και κλιμακώσιμους μηχανισμούς αντιμετώπισης επιθέσεων.

Αρχικά παρουσιάζεται ένας μηχανισμός ανίχνευσης επιθέσεων σχεδιασμένος εξ' ολοκλήρου στο επίπεδο δεδομένων δικτυακών συσκευών. Μέσα από τη γλώσσα P4, υπολογίζονται μετρικές της δικτυακής κίνησης που μπορούν να υποδείξουν άμεσα το θύμα της εκάστοτε επίθεσης. Οι μηχανισμοί που υλοποιούνται στο επίπεδο δεδομένων επιφέρουν ταχύτερους χρόνους ανίχνευσης σε σχέση με τους παραδοσιακούς μηχανισμούς που βασίζονται στο επίπεδο ελέγχου και μπορούν να οδηγήσουν στην καίρια αντιμετώπιση των επιθέσεων.

Ωστόσο, ο εντοπισμός του θύματος αποτελεί μόνο το πρώτο βήμα για την καταστολή μιας επίθεσης, αφού για να γίνει αυτό εφικτό απαιτείται ο διαχωρισμός της δικτυακής κίνησης σε καλόβουλη και κακόβουλη. Συνεπώς, στη συνέχεια της παρούσας διδακτορικής διατριβής προτείνεται ένας καινοτόμος μηχανισμός προστασίας από επιθέσεις που βασίζεται σε χαρακτηριστικά των πακέτων (signatures) και υλοποιείται σε γενικού τύπου εξοπλισμό αξιοποιώντας τις δυνατότητες του framework XDP. Αλγόριθμοι Επιβλεπόμενης Μάθησης αξιοποιούν μόνο τα σημαντικά χαρακτηριστικά των πακέτων και τα κατηγοριοποιούν σε καλόβουλα/κακόβουλα. Για την αντιμετώπιση των επιθέσεων, χρησιμοποιούνται τα ιδιαίτερα χαρακτηριστικά των κακόβουλων πακέτων, όπως αυτά προκύπτουν από μία διαδικασία μείωσης. Συγκεκριμένα, κατασκευάζονται κανόνες αποκοπής που περιγράφουν με όσο το δυνατόν μεγαλύτερη ακρίβεια την εκάστοτε επίθεση χωρίς να επηρεάζουν σημαντικά την καλόβουλη κίνηση. Ο προτεινόμενος μηχανισμός εφαρμόζεται σε δύο μεγάλες κατηγορίες κατανεμημένων επιθέσεων άρνησης παροχής υπηρεσιών, τις volumetric και τις protocol. Η πειραματική αξιολόγηση δείχνει την υπεροχή της συγκεκριμένης μεθοδολογίας έναντι των κλασικών μηχανισμών προστασίας που βασίζονται σε ροές πακέτων: (i) στην ταχύτητα ανίχνευσης

επιθέσεων, (ii) στην κατασκευή βέλτιστων φίλτρων απόρριψης της κακόβουλης κίνησης και (iii) στις αυξημένες επιδόσεις σε ρυθμούς απόρριψης πακέτων.

Τέλος, ολοκληρώνοντας την παρούσα διατριβή επεκτείνουμε τον μηχανισμό προστασίας που βασίζεται σε χαρακτηριστικά των πακέτων σε συνεργατικά περιβάλλοντα αυτόνομων δικτύων. Η συνεργατική ανίχνευση επιτελείται με τη χρήση τεχνικών Ομόσπονδης Μάθησης που επιτρέπουν την συλλογική κατασκευή μοντέλων Μηχανικής Μάθησης χωρίς την άμεση χρήση των προσωπικών δεδομένων των συνεργαζόμενων. Η συνεργατική αντιμετώπιση βασίζεται και πάλι στο framework XDP και προσφέρεται σαν υπηρεσία στους συνεργαζόμενους φορείς δίνοντας τη δυνατότητα για αποδοτική και κλιμακώσιμη απόρριψη κακόβουλων πακέτων. Σε σύγκριση με τις παραδοσιακές μεθόδους συνεργατικής προστασίας, η μεθοδολογία που ακολουθούμε λαμβάνει υπόψιν της τόσο την ιδιωτικότητα των δεδομένων για την ανίχνευση αλλά και την ευελιξία όσον αφορά τους τύπους των κανόνων αλλά και τους υφιστάμενους πόρους.

Αξίζει αναφοράς ότι ο μηχανισμός προστασίας που κατασκευάστηκε στα πλαίσια αυτής της διατριβής δοκιμάστηκε σε πραγματικό δικτυακό εξοπλισμό (έξυπνες κάρτες δικτύου) και οι επιδόσεις του αξιολογήθηκαν βάσει πραγματικών δικτυακών δεδομένων από ετερογενή δικτυακά περιβάλλοντα.

Λέξεις Κλειδιά:

Δίκτυα Οριζόμενα από Λογισμικό, Κατανεμημένες Επιθέσεις Άρνησης Παροχής Υπηρεσιών, Προγραμματισμός Επιπέδου Δεδομένων, Ανίχνευση Επιθέσεων, Αντιμετώπιση Επιθέσεων, Επιβλεπόμενη Μάθηση, Ομόσπονδη Μάθηση

Acknowledgments/Ευχαριστίες

Κατά τη διάρκεια της ολοκλήρωσης της διδακτορικής μου διατριβής νιώθω την ανάγκη να ευχαριστήσω όλους τους ανθρώπους που με βοήθησαν αλλά και μου συμπαραστάθηκαν σε αυτό το δύσκολο και συνάμα εκπαιδευτικό ταξίδι. Άνθρωποι που τόσο με την υποστήριξη σε τεχνικά ζητήματα όσο και με τις πρωτότυπες ιδέες τους αποτέλεσαν ακρογωνιαίο λίθο αυτής της προσπάθειας. Πέραν τούτου, δεν θα μπορούσα να μην συμπεριλάβω και ανθρώπους που στάθηκαν δίπλα μου, με ανέχτηκαν αλλά όποτε και όσες φορές χρειάστηκε μου έδωσαν δύναμη να συνεχίσω.

Θέλω πραγματικά και εγκάρδια να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Βασίλη Μάγκλαρη, για τη συνεχή και ουσιαστική υποστήριξη που μου προσέφερε όλα αυτά τα χρόνια, από προπτυχιακό φοιτητή στο εργαστήριο NETMODE, μέχρι και την ολοκλήρωση των διδακτορικών μου σπουδών. Πέραν όμως των τεχνικών συμβουλών και της καθοδήγησης του, θα ήθελα να τον ευχαριστήσω για την εμπιστοσύνη που μου έδειξε αλλά και τα μαθήματα που μου έδωσε ως άνθρωπο· μαθήματα ανεκτίμητης αξίας που θα με οδηγούν σε όλη μου την ζωή.

Παράλληλα θα ήθελα να ευχαριστήσω τους συνεπιβλέποντες καθηγητές κ. Ευστάθιο Συκά και κ. Νεκτάριο Κοζύρη καθώς και τα υπόλοιπα μέλη της επιτροπής μου για τα εποικοδομητικά σχόλια και τις πρωτότυπες ιδέες τους που συνδιαμόρφωσαν σημαντικά την τρέχουσα διδακτορική διατριβή.

Το ταξίδι αυτό παρότι αρκετά μοναχικό, είχε αρκετούς συνοδοιπόρους που δεν μπορώ να μην τους ευχαριστήσω τον καθένα ξεχωριστά για τη συμβολή τους. Θα ήθελα να ευχαριστήσω ιδιαίτερα τον Δρ. Αδάμ Παυλίδη για την αμέριστη βοήθεια και τις αέναες συζητήσεις μας που με έκαναν να βελτιωθώ τόσο σε τεχνικό όσο και σε ανθρώπινο επίπεδο. Επίσης θα ήθελα να ευχαριστήσω από τα βάθη της καρδιάς μου τον Δρ. Δημήτρη Καλογερά για την άριστη συνεργασία μας και την βοήθεια που μου έδωσε σε πολυποίκιλα επίπεδα. Σημαντική ήταν ακόμη η υποστήριξη της Δρ. Μαίρης Γραμματικού όπως και η συνεργασία μου με τον Δ. Πανταζάτο και Ν. Κωστόπουλο όπου και όποτε χρειάστηκε. Επίσης, θα ήθελα να ευχαριστήσω την κ. Χρύσα Παπαγιάννη για την άριστη συνεργασία μας και τη συμβολή της στη διατριβή μου. Τέλος, θα ήθελα να ευχαριστήσω όλα τα μέλη του εργαστηρίου NETMODE για τις ιδιαίτερα ενδιαφέρουσες συζητήσεις μας αλλά και γενικότερα για όλες τις ωραίες στιγμές.

Ακόμη, θα ήθελα να ευχαριστήσω τους φίλους μου για την αμέριστη υποστήριξη και για όλες τις δύσκολες φορές ήταν εκεί για εμένα, αναπτερώνοντας το ηθικό και την ψυχολογία μου.

Τέλος δεν θα μπορούσα να μην ευχαριστήσω την οικογένεια μου, τους γονείς μου Γιάννη και Μαρία για όλα αυτά που απλόχερα μου προσφέρουν όλα τα χρόνια της ζωής μου, αλλά και τη δίδυμη αδερφή μου, Μαριάνθη, που δίχως δεύτερη σκέψη ήταν και είναι πάντα εκεί για εμένα. Τους αγαπώ και τους ευχαριστώ για την ανιδιοτελή τους αγάπη.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

Acronym	Definition
ACK	Acknowledgement
ACL	Access Control List
AE	Autoencoder
AI	Artificial Intelligence
AM	Anomaly Mitigation
API	Application Programming Interface
AQM	Active Queue Management
AS	Autonomous System
ASIC	Application-Specific Integrated Circuit
ASN	Abstract Syntax Notation
ASR	Aggregation Services Routers
BER	Basic Encoding Rules
BGP	Border Gateway Protocol
BPF	Berkeley Packet Filter
CBN	Communications Blockchain Network
CDN	Content Delivery Network
CLDAP	Connection-less Lightweight Directory Access Protocol
CM	Collaboration Manager
CNN	Convolutional Neural Network
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CSUM	Checksum
DDoS	Distributed Denial-of-Service
DNS	Domain Name System
DPDK	Data Plane Development Kit
DPI	Deep Packet Inspection
ETSI	European Telecommunication Standards Institute
EWMA	Exponentially Weighted Moving Average
EWMD	Exponentially Weighted Moving Difference
FD	File Descriptor
FI	Firewall Instance
FL	Federated Learning
FM	Federated Model
FN	False Negative
FP	False Positive
FPGA	Field Programmable Gate Array
FW	Firewalling
GB	Gigabyte
GDPR	General Data Protection Regulation
GÉANT	Gigabit European Academic Network

GPB	Google Protocol Buffer
GRNET	Greek Research and Technology Network
HTM	Hierarchical Temporal Memory
HTTP(S)	HyperText Transfer Protocol (Secure)
ICMP	Internet Control Message Protocol
ICT	Information and Communication Technologies
ID	Identifier
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPS	Intrusion Prevention System
ISP	Internet Service Provider
IX	Internet Exchange
JSON	JavaScript Object Notation
LDAP	Lightweight Directory Access Protocol
LPM	Longest Prefix Match
LSTM	Long Short-Term Memory
MAC	Media Access Control
MIB	Management Information Base
ML	Machine Learning
MLP	Multi-layer Perceptron
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NF	Network Function
NFV	Network Function Virtualization
NIC	Network Interface Card
NLRI	Network Layer Reachability Information
NMS	Network Management System
NREN	National Research and Education Network
NSGA	Non-dominated Sorting Genetic Algorithm
NTP	Network Time Protocol
NTUA	National Technical University of Athens
OF	OpenFlow
OID	Object Identifier
OOB	Out-of-Bag
POP	Point of Presence
RAM	Random Access Memory
RF	Random Forest
RFC	Requests For Comments
RNN	Recurrent Neural Network
RPC	Remote Procedure Call
RPF	Reverse Path Forwarding
RPKI	Resource Public Key Infrastructure
RR	Resource Record

RTBH	Remotely Triggered Black Hole
SC	Signature Classification
SDN	Software-Defined Networking
SE	Signature Extraction
SLA	Service-level Agreement
SMI	Structure of Management Information
SNMP	Simple Network Management Protocol
SOM	Self-Organizing Map
SR	Signature Reduction
SSDP	Simple Service Discovery Protocol
SVM	Support-Vector Machine
SW	Switching
SYN	Synchronization
TCAM	Ternary Content-Addressable Memory
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TNR	True Negative Rate
TP	True Positive
TPR	True Positive Rate
TU	Thapar University
TW	Time-Window
TWAMP	Two-Way Active Measurement Protocol
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
VM	Virtual Machine
VNF	Virtual Network Function
VPN	Virtual Private Network
WAN	Wide Area Network
WIDE	Widely Integrated Distributed Environment
XDP	eXpress Data Path
YANG	Yet Another Next Generation
ZC	Zero-Copy

1 Introduction

1.1 Motivation & Problem Statement

Internet services have been and still are an integral part of our lives with a plethora of everyday activities being highly dependent on them. These span from applications that facilitate online economic transactions, digital content exchange, social networking, but also extend to critical applications for the human life, e.g. remote surgery. COVID-19 pandemic is one of the recent examples that illustrated the reliance of people on Internet services; during that period, huge Internet traffic growths were observed, since most people were self-isolated spending most of their time at home (e.g. remote working, extent usage of streaming services). This period is a prominent example illustrating that the stability and the guaranteed performance of computer networks can significantly affect our everyday lives.

Network attacks provide the means for disrupting the stable/desired operation of computer networks. Especially, Distributed Denial-of-Service (DDoS) attacks [1] are the most common way for plaguing network infrastructures and overwhelming services offered on top. These attacks aim, using a wide spectrum of techniques, to render specific services and/or network infrastructures unreachable to their legitimate users. DDoS attacks have been commoditized and even offered as-a-service via platforms referred to as *Booters* [2]; in exchange of a small fee, attacks of high volume can be launched, capable to bring down from small enterprises up to large ICT (Information and Communications Technology) companies. The ease of initiating DDoS attacks combined with diverse motives (e.g. extortion, cyber warfare, boredom) have made them an everyday problem for network operators and as a consequence for the legitimate end-users.

The increasing frequency of DDoS attacks impacting critical and of paramount importance Internet services has paved the way for the development of a large DDoS protection industry [3]–[6]. These companies offer full-fledged traffic scrubbing on potential victim networks (e.g. Internet Service Providers, Content Delivery Networks, Academic Institutions) by providing two types of services: they (i) either offer on-demand protection by draining network traffic destined to victim networks, scrub it, and forward back the benign portion, (ii) and/or provide commercial scrubbing appliances [7], [8] for

on-premises protection; these are constantly protecting networks and services from malicious traffic. However, due to the costly fees introduced by commercial-based solutions, custom in-house alternatives are also considered instead. In both cases, DDoS protection frameworks should be able to adapt to the evolving landscape of network attacks and cope with the requirements posed by the ever-growing Internet traffic. Therefore, the protection mechanisms need to consider the following limitations/challenges:

- Proprietary software/hardware - Limited flexibility: Typical DDoS protection solutions are either proprietary in terms of hardware equipment or software implementations. This poses difficulties on managing, troubleshooting or even extending such mechanisms while introducing vendor lock-ins.
- Scalability/Performance: Protection services are required to cope with the ever-increasing Internet traffic. Hardware implementations lack the elasticity of extending on-demand their resources while software-based approaches, although elastic, are not able to meet performance requirements imposed by emerging network infrastructures.
- Adaptability to diverse & complicated traffic patterns: The increase of complicated and diverse Internet applications/services in the Big Data era creates constantly heterogeneous and complex traffic patterns. DDoS protection mechanisms need to deal with the evolving traffic patterns by providing accurate and rapid traffic classification.

Deep network programmability realized by the advent of *Network Softwarization* in combination with the evolution/embracement of intelligent data-driven methods, i.e. *Machine Learning*, can act as key enablers to overcome the aforementioned limitations/challenges.

Network Softwarization was firstly introduced by the *OpenFlow* protocol [9] (and similar efforts [10]) that enabled network operators to program the control plane of their networks in a unified way. Following that paradigm, a new era of programmability has raised awareness, offering high-performance programmable data planes. Two main efforts, P4 [11] and *eXpress Data Path* (XDP) [12], introduced a revolution in computer networks management. Especially for DDoS protection services, these can act as the cornerstone of flexible, scalable, and programmable detection and mitigation pipelines.

Moreover, the evolution of high-performance computing alongside with the establishment of integrated *Machine Learning* frameworks allow the broad use of intelligent data-driven methods, i.e. *Machine Learning*. Traditional statistical approaches may be inadequate to analyze the diverse and complex patterns of network traffic. Thus, *Machine Learning* is a promising candidate for accurate, adaptable, and automated traffic classification.

The aforementioned technologies can play a significant role in addressing DDoS protection research challenges. These are divided in two discrete but non-independent categories: DDoS (i) detection and (ii) mitigation. DDoS detection includes mechanisms for network data extraction and analysis towards the identification of (i) ongoing attacks, (ii) targeted victims, (iii) attack types, and (iv) malicious traffic portions. The key performance indicators of these tasks are the immediacy (in terms of time) and accuracy. The former affects the countermeasures reaction time while the latter the legitimate users' quality of experience. DDoS mitigation includes methods/techniques to effectively filter out malicious traffic without impacting benign traffic. Scalability, flexibility, and performance are the key challenges to be considered by DDoS mitigation solutions.

1.2 Contributions

Based on the aforementioned challenges and innovative technologies, in this dissertation, we leverage on recent advances in computer networks and intelligent data-driven algorithms to architect an integrated scalable, fast, adaptable, and efficient DDoS protection mechanism. Our key contributions in comparison to the existing state-of-the-art approaches are summarized below:

Accurate & Rapid DDoS Detection Offloaded in the Data Plane: Contrary to typical control plane traffic monitoring and DDoS detection mechanisms (based on *sFlow* [13], *NetFlow* [14] or *OpenFlow* [9]), we introduce a rapid detection mechanism in the data plane. Especially, P4 language [11] enables us to design and implement line-rate data plane pipelines that can accurately detect network anomalies. DDoS attacks are identified within short timeframes providing the means for fast remediation of the anomaly.

Intelligent Data-Driven Signature-based Traffic Classification: Traditionally, DDoS protection mechanisms classify network traffic based on packet data organized in network flows. This poses difficulties with regards to collection, processing, and storage hindering

real-time detection and mitigation. Unlike flow-based schemes, we employ packet signatures that instantly reveal DDoS traffic characteristics. These are identified via *Machine Learning* models providing rapid, automated, and adaptable traffic classification.

Source-IP agnostic DDoS Mitigation driven by Smart Filtering Rules Reduction: Filtering rules are commonly applied in commodity network devices (switches, routers, firewalls) that impose limits to the number of entries they can support. To reduce their number, source-IP based filtering schemes employ aggregation techniques by organizing malicious IP addresses in subnets. In contrast, we introduce a filtering rule reduction mechanism tailored to the attack traffic characteristics. This identifies a concise set of filtering rules able to filter out the attack traffic, with minimal effect on benign traffic.

High-performance Scalable Network Functions based on Programmable Middleboxes: In legacy network environments, traffic monitoring and filtering are implemented in rigid proprietary appliances. In contrast, we opted to use programmable COTS (Commercial off-the-shelf) hardware (i.e. low-cost NICs) powered by the XDP framework. This enables the design and implementation of Virtual Network Functions (VNFs) that can be instantiated on-demand and scaled according to traffic and application requirements, thus suitable for elastic scrubbing services.

Privacy-preserving DDoS Detection and Scalable Mitigation tailored to Collaborative Network Environments: Autonomous Systems (AS's) collaborations are instrumental in the Internet success story, but this is largely not extended to attack protection. Collaborative DDoS detection is hindered by strict data privacy legislations while mitigation by rigid firewall solutions. To address such concerns and limitations, we introduce a signature-based DDoS protection framework tailored to collaborative network environments. DDoS detection is performed in a privacy-preserving fashion via the *Federated Learning* technique and DDoS mitigation is offered to collaborating parties as a flexible/scalable service.

Experimentation using Production Network Data on Real Computing and Network Hardware: We employ real computing and network resources to conduct high performance experiments assessing the applicability of the developed mechanisms in realistic network environments. Our experimentation is based on network traces from

production network environments, i.e. Campus networks, Internet Service Providers (ISPs) and Internet Exchanges (IXes), thus allowing us to evaluate our methods/algorithms using both real and heterogeneous network data.

1.3 Outline

The remainder of this dissertation is structured as follows:

Section 2 provides a brief overview of computer networks and their evolution to meet the ever-increasing needs imposed by Internet advances. Initially, we briefly discuss computer networks and their operational characteristics; subsequently, the *Network Softwarization* paradigm is introduced covering the evolution of Software-Defined Networks from *OpenFlow* (OF) to programmable hardware (P4) and software data planes (*eXpress Data Path*).

Section 3 presents concepts and technologies related to network monitoring. Monitoring protocols, techniques, and data are investigated both for legacy and programmable network environments. We put an emphasis on data that can be exported from network devices and focus on their use for anomaly detection tasks.

Section 4 introduces the problem of Distributed-Denial of Service (DDoS) attacks and analyzes the different attack types with a focus on their specific characteristics. Subsequently, state-of-the-art DDoS attacks detection mechanisms/algorithms are elaborated and finally mitigation techniques are discussed.

Section 5 explains our work on P4-based DDoS attack detection. The proposed approach attempts to address the problem of DDoS detection entirely in the data plane providing rapid and accurate coarse-grain DDoS alerts (pinpoints anomalies for hosts/subnetworks). Our mechanism is evaluated on network hardware (programmable P4-enabled Network Interface Cards) using production network data.

Section 6 makes a step forward towards DDoS protection. We propose a framework that attempts not only to detect DDoS attacks but also to classify and filter malicious traffic. Specifically, we consider SYN Flood attacks (as an indicative use case of protocol attacks) and use packet signatures to classify and filter them. Our approach leverages on *Machine Learning* techniques for traffic classification and softwarized data planes for

efficient packet filtering. We use captured network attacks to compare our schema to the state-of-the-art mitigation approach *SYN Cookies*.

Section 7 introduces a generic signature-based classification and filtering scheme for volumetric attacks, extending the concept presented in section 6. The proposed framework identifies the most important packet features for traffic classification and generates IP-agnostic filtering rules for effective packet filtering. Our approach is thoroughly evaluated against state-of-the-art source IP/flow-based approaches using real production network data.

Section 8 extends the work presented in sections 6, 7 on signature-based DDoS protection to collaborative multi-domain network environments. We leverage on the *Federated Learning* paradigm to detect DDoS attacks in a privacy-aware fashion and design a scalable and programmable DDoS mitigation as a service tailored to collaborative network environments. Our schema is evaluated on multi-domain production network data.

Section 9 summarizes the contributions of this dissertation and proposes future steps and directions on open problems with regards to DDoS attack protection.

Section 10 provides an extended abstract of this dissertation in Greek, **Section 11** provides author's publications and finally **Section 12** contains references/bibliography.

2 Computer Networks & Network Programmability

2.1 Computer Networks

Computer networks are and have always been the core ingredient of Internet infrastructures, enabling for user and service interconnection. Network devices (routers, switches, firewalls etc.) are the cornerstone of Internet infrastructures; these are used first and foremost for transferring information between users and services but also to protect them from malicious actors. In legacy network environments, packet forwarding is based on management/control decisions determined by each network device. Specifically, the network operations are categorized in the following planes (described below in a top-down approach):

Management Plane: The management plane embeds all the operations related to computer networks configuration and monitoring. The former may span from security policies for network devices protection to control plane configurations (e.g. routing protocols). The latter refers to network data collection and analysis that are useful for maintaining the desired state of networks while validating their proper functionality.

Control Plane: The control plane defines switching/routing rules on network devices based on switching/routing processes; these rules are applied on packets as they traverse network devices and determine the way packets are forwarded in computer networks.

Data/Forwarding Plane: The data plane processes network packets in real-time and applies the desired logic, specified by control/management operations. Indicative data plane operations include packet switching (destination port selection) based on MAC addresses, packet routing based on destination IP addresses and packet filtering based on Access Control Rules.

In legacy computer network architectures, data, control and management planes are intertwined at the device level; such an approach was suitable for network management and operational processes in legacy environments. Emerging technologies evolved in the information and communication technology (ICT) domain, in particular, 5G, Cloud Computing, Big Data, Network Function Virtualization (NFV), Internet of Things (IoT), and Intent-based networking explicit the need of high bandwidth, ubiquitous accessibility, and dynamic management of computer networks [15].

This explosion of Internet services, revealed new traffic requirements that legacy network architectures were unable to cope with. Key considerations were related to: (i) limited network/device programmability, (ii) the absence of open management standards and vendor lock-ins, and (iii) the complexity of network infrastructures posing management difficulties. These considerations led to a revolution in computer networks emerged by the Software-Defined Networking (SDN) paradigm.

2.2 Software-Defined Networks

The aforementioned considerations in parallel with advances in network hardware and software drove researchers and operators to rethink traditional network architectures. Key design principles for next-generation networks were deep network programmability, open and standardized interfaces for unified network management.

2.2.1 OpenFlow Protocol

OpenFlow (OF) [9] is considered one of the first and well-established protocols of the Software-Defined Networking (SDN) paradigm. OF created the pathway for innovative network (SDN) architectures by disaggregating the control from the data plane. In a nutshell, the purpose was to transfer the "intelligence" of computer networks from the network device to centralized controllers, as shown in Figure 2.1. This architecture provides (i) a wide centralized view of the network substrate, (ii) optimized performance through centralized decisions, and (iii) granular network-wide policy configuration and management.

OF originally defined the communication protocol in SDN architectures that enabled external controllers to directly interact with the forwarding plane of network devices (switches, routers). The forwarding plane of OF-enabled devices consists of match/action tables that contain (i) a set of rules (based on packet fields spanning from L2 to L4) that match traversing packets and (ii) a set of possible actions, e.g. forward to specific port, drop the packet. These rules can be dynamically programmed by applications via OF's unified interface. OF was widely employed by researchers for various network applications considering use cases for network security [16], [17] but also by production

environments, e.g. in Google they designed a flexible and elastic software-defined Wide Area Network (WAN) [18].

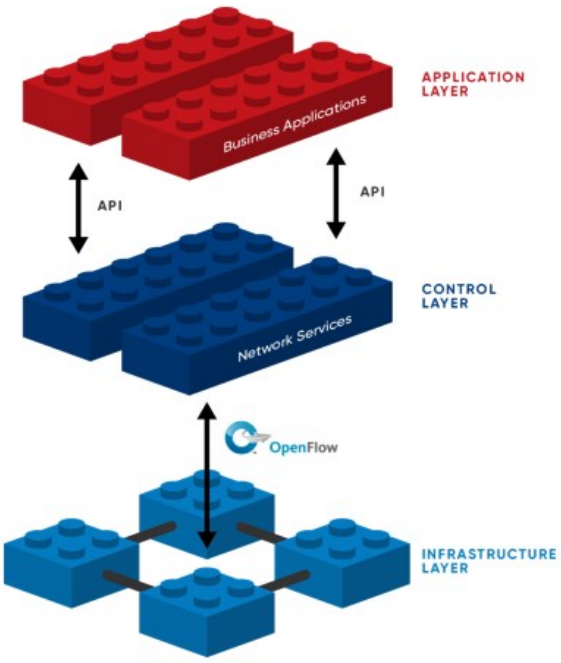

Figure 2.1: *OpenFlow* application, control, and infrastructure layer interactions [19]

2.2.2 Hardware Data Planes

Although OF created new pathways for programming network devices, "*OpenFlow main goal was to make it easier for those who own and operate networks to write better control planes*.", as N. McKeown mentions. OF was based on the hypothesis that switch chips are not programmable and attempted to fill the gap of unified programming interfaces across network devices. However, from recent advances in network hardware, programmable switch chips were designed that can achieve comparable performance to the typical fixed-function chips. This revealed new capabilities on programming network devices as their data plane could be directly programmed by specifying the journey of packets within the hardware pipeline. In a similar fashion with OF, the need for a common way to program data planes was required. Therefore, in 2014, a group of researchers introduced P4 (Programming Protocol-independent Packet Processors) language [11], a domain-specific language allowing developers to abstractly express packet forwarding logic and apply it directly to network devices.

A consistent effort that followed the development of P4 language was the evolution of programmable packet processors, e.g. DPDK [20], XDP [12]. Softwarized programmable

data planes were incorporated in Linux systems presenting high packet processing capabilities. Programmable packet processors allow developers to program high-performance applications on COTS Network Interface Cards (NICs). Softwarized data planes were mostly embraced by key players of the ICT industry [21] to design and implement scalable, flexible, and of high-performance applications [22]–[24].

Both approaches (hardware data planes, software data planes) introduce the in-network computing paradigm [25] that enables offloading computing tasks (e.g. Network Functions) in programmable but of high-performance data planes; this creates a new surface for developing novel network applications suitable for use cases that require rapid decision making, e.g. anomaly detection tasks. More details related to the P4 framework and its architecture are provided in subsection 2.2.2.1; details about programmable packet processors and especially for the XDP framework are presented in subsection 2.2.3.

2.2.2.1 Programming Protocol-independent Packet Processors (P4)

P4 [11] is a high-level language for expressing how packets are processed by the data plane of programmable network devices (switches [26], NICs [27]). The core design principles of P4 are:

- *Reconfigurability*: Network devices forwarding behavior should be able to be re-programmed on the fly depending on the network application.
- *Protocol Independence*: Network protocols change/evolve to meet new requirements; adding new or extending protocols should be able to be programmed on-demand without involving timely procedures (long lifecycles of vendors).
- *Target Independence*: Network devices should be able to be programmed in a common way regardless of the specifics of the underlying hardware.

2.2.2.2 P4 Overview

In Figure 2.2 below, the lifecycle of deploying P4 programs at network devices (targets) is depicted. Device manufacturers provide the hardware or software implementation framework, an architecture definition, and a P4 compiler for that target. P4 programs are written for a specific architecture (P4 architecture model), which defines a set of P4-programmable components on the target as well as their external data plane interfaces [28]. The compilation of a P4 program generates (i) a data plane program tailored to the

employed hardware and (ii) an API that exposes read/write functionalities between the control and the data plane. In the next subsection, we will focus on the basic primitives of P4 programs development.

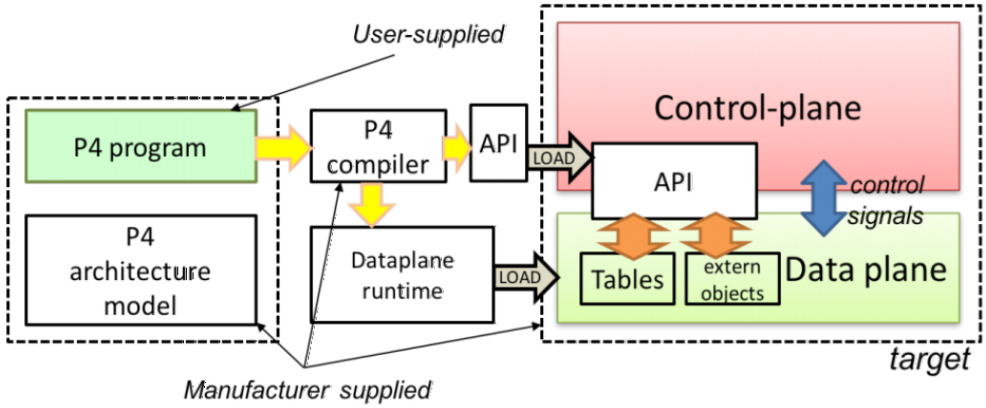

Figure 2.2: Programming a network device (target) with P4 [28]

2.2.2.3 Architecture Model

The P4 architecture model is a crucial component for the development of P4 programs. It is a reference model that defines the programmable blocks of P4-enabled devices and their data plane interfaces. We will describe the *v1model* architecture to explain the components of a typical P4 architecture and the programming capabilities it offers (*v1model* was used as a reference architecture for designing a P4-based DDoS detection scheme presented in section 5). Note that, *v1model* is a well-established architecture model used in software switches, i.e. BMv2 [29] but also supported by NICs manufacturers [27].

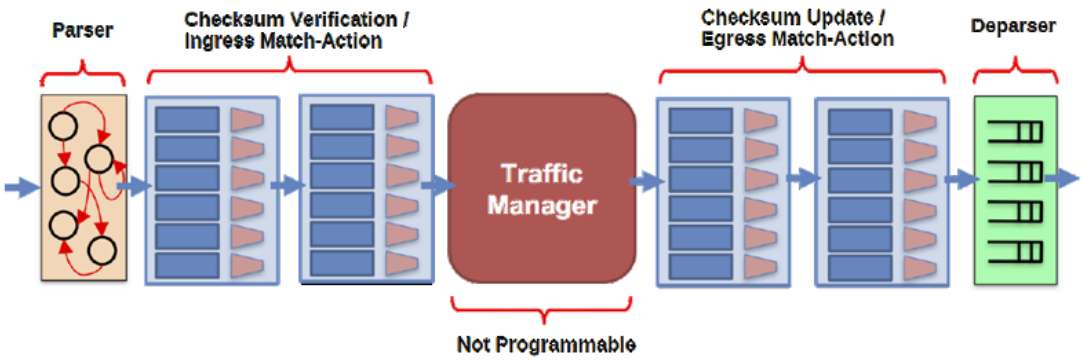

Figure 2.3: P4 *v1model* architecture

v1model architecture consists of a 6-stages data plane pipeline as depicted in Figure 2.3. The *Parser* defines all the available packet headers that are supported by the P4 program and the order of packet header parsing (e.g. from L2 to L4). Subsequently, the packet

passes to the *Checksum-Verification* stage, in which packets may be verified for corrupted packet headers. The *Ingress Match-Action* stage follows; typically, in that stage, the forwarding logic is applied (switching/routing) and the packet is assigned to the traffic manager (not programmable in the *v1model* architecture). The next stage is the *Checksum Update* stage that updates the checksum values of the packet which may be necessary due to altered packet fields on previous stages, e.g. in *Ingress Match-Action*. An *Egress Match-Action* stage follows, in which similar logic with the *Ingress Match-Action* table can be applied, and finally, the packet is passed to the *Deparser* stage, which emits the packet to the appropriate outgoing port.

2.2.2.4 P4 Programming

P4 programs are developed based on the architecture model supported by the manufacturer of the network device. P4 language provides a set of capabilities that illustrate similarities with common programming languages such as C. We describe below in detail the journey of a packet from the arrival in a network interface up to the departure from the P4-enabled device. In parallel, we also present the basic capabilities offered by the P4 language, describing its basic primitives (using *v1model* as the reference architecture model).

The first stage, that packets are processed, is the *Parser,* in which developers define the packet headers that can be employed subsequently by the P4 program. Packet headers are *structs* that include simple data types (e.g. *integers*, *bits*, *booleans*); these can be used for the development of any network protocol using just a few lines of P4 code. After the definition of the available packet headers, an hierarchical tree structure with the possible packet headers combinations is defined. Incoming packets are matched to the defined packet headers in the *Parser* stage and the packet header values can in turn be used in the next stages of the P4 pipeline.

The most important stage with regards to the desired logic of P4 programs is the *Ingress Match-Action* stage. This may incorporate combinations of the following primitives of the P4 language:

- Mathematical operations: These include simple operations like additions, multiplication, and bit shifting and can be applied to the values of packet fields.

- Packet metadata: These may be categorized in (i) User-defined metadata and (ii) Intrinsic metadata:
    - User-defined metadata: These describe data structures defined by the programmer and are per-packet data, transferrable between the stages of P4 pipelines.
    - Intrinsic metadata: These are also defined at a per-packet level but correspond to special metadata provided by the architecture, e.g. packets' input port.
- Match-action tables: These tables are similar to key-value stores. The key may be an arbitrary combination of packet metadata and/or headers while the value is associated with an action; actions are functions that set the packet metadata/header values. Note that match-action tables are defined in the data plane but are only populated by control plane functions.
- Registers: These are also key-value stores (similar to single dimension arrays) that can be set both by control plane functions but also in the data plane. These are extremely useful for designing algorithms in the data plane that require per-packet state information.
- Extern functions: These are special-purpose third-party functions offered by the underlying architecture/target. Indicative examples include hash functions and high-accuracy timestamping.

Based on the aforementioned capabilities and constraints of the P4 language a wide spectrum of Network Functions can be implemented in the *Ingress Match-Action* stage. These span from simple forwarding tasks, i.e. packet switching (selecting the outgoing port from a match-action table based on the destination MAC address) to complex algorithms such as Heavy-Hitter detection [30], DDoS detection [31], [32], and Active Queue Management (AQM) schemes [33]. Note that, the *Egress Match-Action* stage can also be employed for Network Functions implementation using the same primitives as the ones mentioned for the *Ingress Match-Action* stage.

2.2.3 Software data planes - eXpress Data Path (XDP)

Implementing services in hardware data planes enables for low-latency and high-throughput, due to the native performance of switching Application-Specific Integrated Circuits (ASICs). With the advent of Network Function Virtualization (NFV) [34],

Network Functions (NFs), that were naturally operating at physical network appliances, were transformed to software-based solutions (Virtual Network Functions – VNF). This paradigm was initiated from service providers and mobile network operators in an attempt to decouple traditional NFs, e.g. Network Address Translation (NAT), Firewalling (FW), and Deep Packet Inspection (DPI), from proprietary hardware and instead substitute them with software-based solutions on COTS equipment. However, performance implications were expected after replacing hardware-based services/functions with software-based. To that end, Programmable Packet Processors came to the surface, that allow COTS equipment, such as programmable NICs, to achieve comparable performance to expensive ASICs, but with greater capabilities in terms of flexibility and programmability.

Data Plane Development Kit (DPDK) [20] is probably the most well-known framework for programmable packet processing in Linux systems. Although it was initiated as an *Intel*'s endeavor, currently it is supported by many NICs manufacturers. DPDK is a kernel-bypass framework, that removes the control of the networking hardware from the Linux kernel and transfers it to the networking application (bypasses the Linux kernel). A similar approach that also bypasses the Linux kernel is the PF_RING ZC framework [35]. Kernel bypass is a promising approach for developing high-performance VNFs [12], [36], [37] however, due to the non-involvement of the Linux kernel, it has significant management, maintenance, and security drawbacks [12].

An alternative approach for programmable packet processing is the *eXpress Data Path* (XDP) [12], which harmonically co-exists with the Linux kernel. XDP is executed before heavy networking stack operations and can be seamlessly ported in Linux systems. It provides high-performance programmable packet processing in COTS hardware, thus enabling for the deployment of demanding network applications even within legacy servers. In this dissertation, we employed XDP to design and implement high-performance yet programmable monitoring and filtering mechanisms for DDoS detection and mitigation tasks. XDP has been widely adopted in production network environments for various applications, e.g. Load-Balancing [23], Intrusion Detection [24], and DDoS protection [22].

2.2.3.1 XDP Design Principles

We present below the core design principles of the XDP framework:

- Coexistence with the existing Linux networking stack & transparency to user-space applications: XDP can coexist with Linux networking stack while being transparent to applications running on hosts. This enables innovative deployment scenarios such as inline protection against denial of service attacks on commodity servers.
- Programming of multi-vendor NICs in a unified way: XDP programs can be deployed in different (multi-vendor) NIC drivers; there are no special hardware features required, only the existing drivers to be modified for supporting XDP execution hooks.
- Reusing of existing Linux kernel network stack features: XDP allows utilizing Linux network stack features such as the routing table and the TCP stack. This enables developers to focus mostly on the desired functionality of XDP programs without needing to recreate core functionalities of common network applications.
- Online reprogrammability and on-demand scaling: Applications programmed in XDP, can be dynamically reconfigured without any service interruption. Desired features can be added on the fly or removed completely when they are not needed without network traffic interruption. Depending on the traffic loads received by XDP programs, dynamic scaling of the CPU resources (within a single server) may be considered.

In the following subsection, we will delve into details related to the XDP programming model, analyzing in detail the practical aspects of the aforementioned design principles.

2.2.3.2 Programming in XDP

XDP programs, written in C, are executed either in software within the context of the network driver or even offloaded directly in Network Interface Cards (NICs), e.g. *Netronome* SmartNICs [27]. Their execution is initiated upon the arrival of packets at a network interface. In turn, packet data can be parsed, extracted, and stored in persistent memory referred to as *Berkeley Packet Filter* (*BPF*) *Maps* [12] (see Figure 2.4). These are key-value stores defined when the XDP program is loaded. XDP returns an action for each packet which defines how it should be handled. The packets can be either (i) dropped - XDP_DROP, (ii) passed to the network stack - XDP_PASS, (iii) redirected to another interface - XDP_REDIRECT or (iv) transmitted back - XDP_TX.

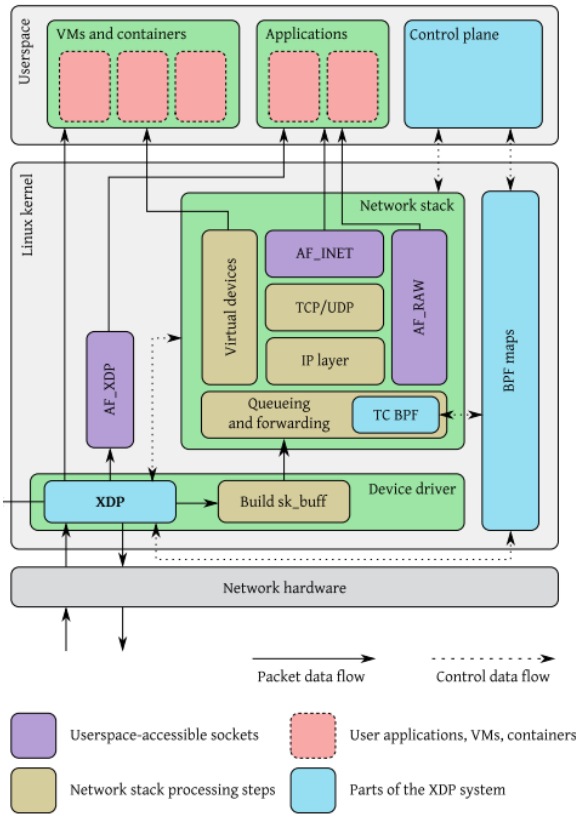

Figure 2.4: XDP integration with the Linux network stack [12]

XDP programs are running in the kernel address space and thus can access (and potentially alter) Linux kernel's memory. For safety purposes, XDP programs before being loaded are analyzed by the *eBPF Verifier*; this component checks XDP programs memory accesses while ensures that the program will terminate. These checks are performed to guarantee that the user-supplied XDP program will not affect the operational status of Linux servers, e.g. kernel malfunction, however, they pose significant challenges on XDP applications implementation; indicative limitations include (i) bounded loops, (ii) fixed-size data structures, (iii) 4096 *BPF* instructions per program, and (iv) limited support of kernel functions. To that end, the design and implementation of XDP applications require significant attention due to the aforementioned limitations. In sections 6, 7, and 8, we discuss the limitations and challenges we faced on developing XDP-based monitoring and filtering components.

3 Network Monitoring

Network operators configure and manage network infrastructures while receiving feedback from them via retrieving network monitoring data. Network monitoring is crucial for network management as it provides information related to the status of network infrastructures (health) and can be used to validate the desired operation/state, commonly driven by (pre-agreed) business requirements. Network monitoring includes a wide spectrum of technologies that are used to export information from network devices. These technologies follow the evolution of network infrastructures attempting to meet the ever-growing requirements for accurate, reliable, and real-time network monitoring.

3.1 Simple Network Management Protocol

Simple Network Management Protocol (SNMP) was and may still be the flagship of network monitoring. SNMP is used to collect information from network devices in a client-server architecture. Typically, monitoring architectures include centralized Network Management Systems (NMS) that periodically poll network devices (agents) requesting information about their current status. The available information is defined in hierarchical data structures, referred to as Management Information Bases (MIBs). Each object in the MIB is identified by a unique Object Identifier (OID) and corresponds to data related to the network device; these may be either retrieved or modified. Available data provided by network devices are highly dependent on the existing MIBs, which may be either proprietary (vendor-specific) or standardized.

SNMP is still used by network operators aiding them to detect, identify, and solve problems that occur in their networks. In typical use cases, centralized/distributed collectors request data (e.g. interfaces bandwidth utilization, device status) from network devices; these data can be employed for multiple purposes ranging from applications for DDoS detection [38] to network design procedures, e.g. capacity planning. Although SNMP seems an ideal protocol for managing and monitoring networks, it has plenty of limitations. SNMP proved to be inadequate for providing monitoring data in modern large-scale infrastructures [39]. Increased polling times (5-minute intervals), data collection scalability issues [40], unreliable delivery (UDP as the transport protocol) are only some of the drawbacks that forced network device vendors and operators to move towards different monitoring solutions, i.e. Streaming Telemetry.

3.2 Streaming Telemetry

"*Streaming telemetry is a new approach for network monitoring in which data is streamed from devices continuously with efficient, incremental updates*"[1]. Streaming Telemetry mechanisms overcome limitations imposed by SNMP. Specifically, the data collection process does not rely on polling-based schemes but on a push-based/streaming fashion, allowing devices to send information to external collectors even upon data change. Reliable delivery is ensured via TCP while authentication/authorization is based on user/password schemes and/or TLS certificates. Data models (similarly to SNMP) can be vendor-neutral or vendor-specific and formatted in *Yet Another Next Generation* (YANG) models commonly serialized via highly compressed mechanisms, e.g. *Protocol Buffers* [41]. The main differences between SNMP and Streaming Telemetry mechanisms are summarized in the following table:

Table 3.1:SNMP vs Streaming Telemetry

	SNMP	**Streaming Telemetry**
Collect Model	Poll	Push - Stream
Transport Layer	UDP	TCP
Application Layer	SNMP	HTTP gRPC
Data Model	MIB proprietary/standardized	Vendor-specific/neutral
Data Format	SMI / ASN-1	YANG
Encoding	BER	Google Protocol Buffers (GPB) or JSON
Security	Communities or Keys (v3)	User/Password or TLS certificate

[1] https://www.openconfig.net/projects/telemetry/

Network monitoring architectures include a centralized collection engine (SUBSCRIBER) that retrieves data from network devices (PUBLISHER). There are two ways for subscribing to data: (i) DIAL-IN and (ii) DIAL-OUT. In the former, the collector subscribes to the data of interest, e.g. CPU utilization while in the latter data subscription is configured in the network device. DIAL-IN provides a flexible, scalable, and dynamic network monitoring approach while DIAL-OUT requires each network element to be configured for the data types that is going to populate to preselected collectors. Indicative data, that can be retrieved from network devices, may be found in [42] (for *Juniper* devices).

Monitoring approaches relying on either SNMP or Streaming Telemetry usually gather information from network infrastructures related to their current state. This state includes information spanning from the current status of network interfaces/devices to complex service monitoring operations (e.g. TWAMP measurements [43]). As mentioned, this information is of paramount importance for network administrators aiding them to successfully monitor and manage their networks. However, such approaches do not provide insight into the actual network data, i.e. network packets/flows that traverse Internet infrastructures. In the following subsections, we discuss *NetFlow* [14] and *sFlow* [13] that provide packet and flow-level information of the network traffic exchanged between users/services.

3.3 NetFlow

NetFlow is a network protocol introduced by *Cisco* that enables network operators to gain insight into the network traffic sourced/destined from/to their network via the representation of network flows. A network flow, defined by the 5-tuple (source IP address, destination IP address, source port, destination port, protocol), provides information about the packets that were exchanged between endpoints/services within a specific time interval by aggregating related data, e.g. packet/bytes counters. This process is continuously conducted in network interfaces (either examining each packet or picking 1 out of n samples – sampled *NetFlow*) as packets traverse network devices. Flow data are stored temporarily in the flow caches of network devices for preconfigured time intervals (based on active/inactive timeouts) and upon their expiry conveyed to external collectors (see Figure 3.1). These typically store data related to the observed flows that can be subsequently used for further analysis. Network administrators may use flow

information for network management tasks, e.g. network monitoring/troubleshooting, network capacity planning, customer billing, and/or network anomaly detection tasks. Especially for the latter, *NetFlow* has been and is still widely used for detecting and identifying DDoS attacks; this will be further discussed in section 4.

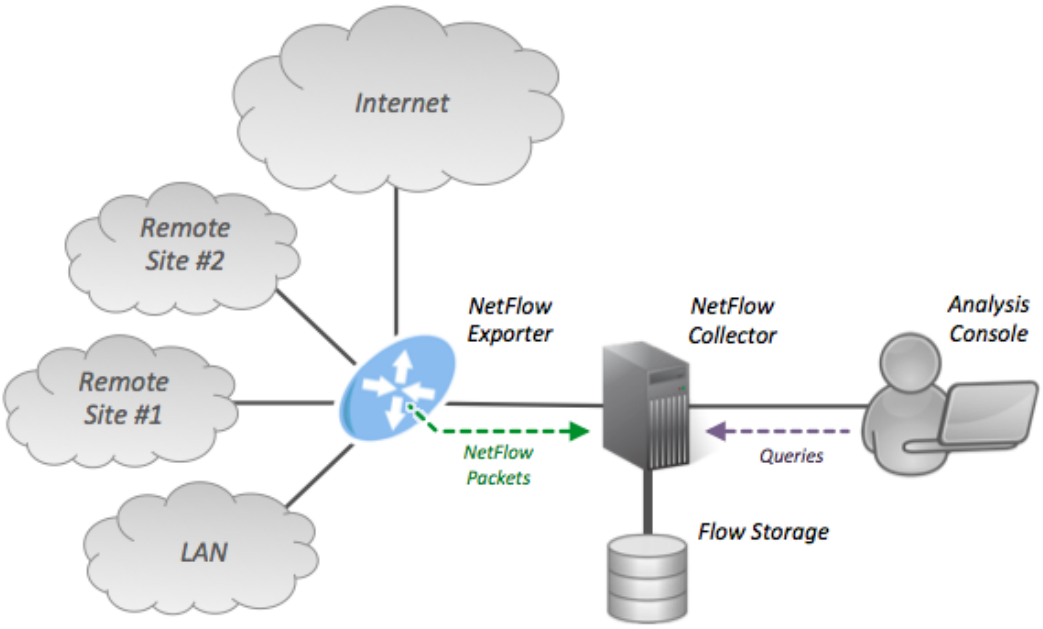

Figure 3.1: *NetFlow* Architecture [44]

3.4 sFlow

sFlow stands for "sampled flow" and is an industry-standard mechanism for extracting packets from network devices at the data link layer. This mechanism allows network devices to push data (packet samples and/or interface counters) to external collection engines which can employ them for network monitoring operations. *sFlow* is typically configured with sampling rates based on the interface speed from which network packets are sampled. Although sampling appears as a limitation, in reality, *sFlow* is a scalable mechanism for network monitoring in high-speed switched or routed networks. This is validated from different use cases reported in the literature, e.g. network anomaly detection [16] but also from production environments, e.g. Cloudflare's DDoS protection framework [45].

The main characteristic of *sFlow* is that it gives access to packet (i) headers and (ii) payload. Packet headers can be used to aggregate packets in network flows in a similar fashion to *NetFlow*. In contrast, packet payload can be used directly for identifying

anomalies in packet data, e.g. malicious pattern identification. In section 4, we will discuss in detail how data provided by *sFlow/NetFlow* may be employed as data sources for DDoS detection pipelines.

3.5 Deep Packet Inspection

Deep Packet Inspection (DPI) refers to the process of inspecting the contents of all network packets that traverse network devices. Network packets should be first copied and redirected (e.g. via port mirroring or monitoring taps) to Deep Packet Inspectors that capture and analyze them. DPI may be used for various purposes: to baseline application behavior, analyze network usage, troubleshoot network performance, data validation, malicious code checks or DDoS attack detection. Especially in DDoS attack detection/prevention tools such as *Snort* [46] and *Suricata* [24], packets are compared against a set of rules (signatures) that correspond to pre-identified anomalous packet patterns.

Deep Packet Inspection may be an intensive process both for the system that collects network packets but also for the network elements that copy the desired network streams. However, it may reveal packet characteristics that may not be available via the aforementioned network monitoring methods, i.e. unobserved packets due to sampling.

3.6 Software-Defined Networks

3.6.1 OpenFlow

Monitoring OF-enabled networks provides a greater flexibility on the available information that could be exposed by network devices. As mentioned, OF uses flow tables that may include large numbers of packets fields. Each rule in the flow table is accompanied by network statistics (packets, bytes counters). This allows to retain aggregate data for arbitrary combinations of packet fields beyond the well-known 5-tuple (network flow). Customizing the monitoring data tailored to network applications is an appealing concept, however there are two major drawbacks. In OF, the forwarding logic is tightly coupled with network monitoring [16] and thus network data are only available for aggregations that have been included in the flow table due to the forwarding logic. This may present scalability limitations in case of large networks due to the massive number of packet field combinations that can be in parallel in the flow table [16]. The

second drawback is related to the fixed set of packet fields exposed by the device. This is highly related to the supported OF protocol version and the corresponding vendor implementation. Requesting new packet headers requires vendors' intervention, a timely procedure.

3.6.2 Programmable Data Planes

With programmable data planes, the drawbacks presented in OF environments can be overcome. Interestingly, in both hardware (P4) and softwarized (XDP) data planes mentioned in Section 2, network monitoring is disaggregated from the forwarding logic. This enables to monitor and collect fine-grained data disassociating the forwarding application from the monitoring logic. Except for this disaggregation, the holy grail of programmable data planes is the definition of the exact data that required to be monitored for each application. This simplifies network data storage and analysis as only data of interest are exported. Moreover, it allows to rapidly modify the monitored data adapting to possible protocol extensions or newly introduced network applications.

In the following section, we delve into the main focus of this dissertation, the detection and mitigation of Distributed Denial-of-Service (DDoS) attacks. We (i) discuss their main characteristics, (ii) investigate useful monitoring data for network traffic analysis, (iii) analyze algorithms/methods for attack detection, and (iv) finally present mitigation techniques.

4 DDoS Attacks – Detection & Mitigation

Distributed Denial-of-Service (DDoS) attacks still present a major threat faced by network operators [47]. These attacks plague network infrastructures by overwhelming their capacity and/or processing resources rendering them unable to serve legitimate users. DDoS attacks are typically orchestrated by malicious actors, e.g. hackers, that command bots (infected hosts) to generate malicious traffic targeting selected services and/or hosts, as depicted in Figure 4.1 below.

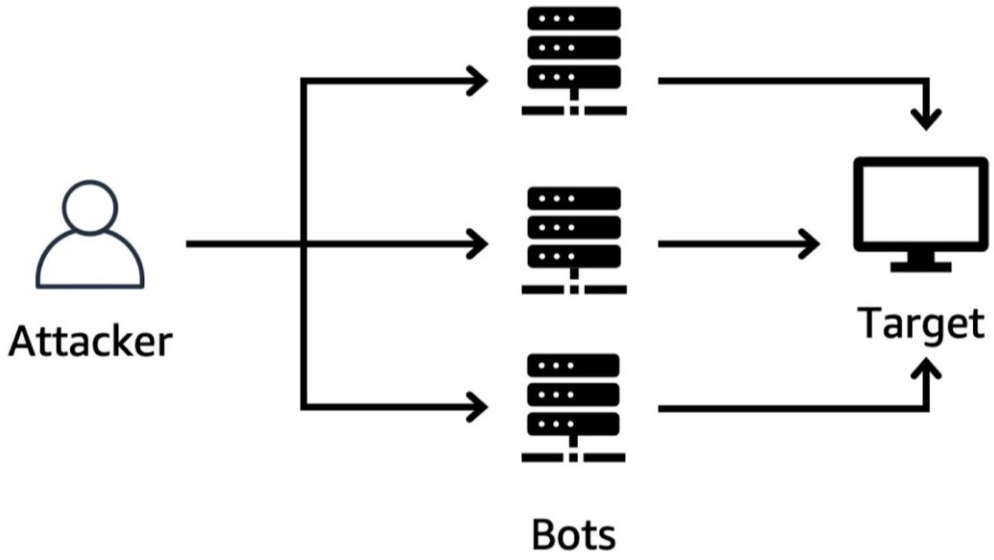

Figure 4.1: Distributed Denial-of-Service Attacks Orchestration[2]

These bots (or zombies) are typically vulnerable devices with IP connectivity compromised by malicious actors and employed not only for DDoS attacks but also for other malicious purposes, e.g. port scanning, email spam campaigns.

DDoS attacks are categorized in three different attack types, each one with different characteristics; however, they all serve the same purpose, to harm the selected victim network/service. In the next subsection, we discuss in detail the different attack types.

[2] https://trailhead.salesforce.com/en/content/learn/modules/aws-cloud-security/protect-against-dos-and-ddos-attacks-with-aws-shield

4.1 Attack Types & Characteristics

We may categorize DDoS attacks, based on the way they disrupt network infrastructures/services, in the following types: (i) volumetric, (ii) protocol, and (iii) application-layer attacks.

Volumetric attacks create link congestion by consuming all available bandwidth between the targets (victims) and their upstream providers/peers. Enormous amount of data is sent to victim networks either via amplification techniques or other means of massive traffic generation. A typical example of such attacks is the Reflection/Amplification case, in which attackers exploit vulnerable protocols and services to generate attack traffic (their magnitude is measured in bits per second - bps). Attackers use the IP address of the selected victim and send specially crafted requests to "misconfigured" servers (reflectors). These respond to the falsified requests with packets of huge payload that consume victims network bandwidth. Note that a common side effect of such attacks is packet fragmentation since large responses, generated by reflectors, typically exceed the Maximum Transmission Unit (MTU) of transit links. Commonly exploited protocols/services for volumetric attacks include DNS, SNMP, CLDAP, NTP and SSDP [48].

Protocol attacks disrupt network services by overwhelming the resources of end-hosts and/or the resources of interim network devices (e.g. firewalls and load balancers). These attacks exploit "vulnerabilities" of the network and/or transport layer to increase the processing burden of the selected targets. Specifically, the selected victims constantly attempt to keep state information related to received requests (that may be even spoofed). This results to excessive resource consumption, preventing them from serving legitimate requests. These attacks are typically measured in packets per second – pps (since each packet increases the burden for the victim) and include a wide variety of techniques. Indicative attacks that exploit the 3-way handshake of TCP are SYN [49], ACK [50], SYN-ACK [51] floods. Similarly, ICMP and UDP packets are commonly used to flood victims and force them to waste their resources on responding to falsified/random requests.

Application layer attacks, commonly referred to as layer 7 attacks, exhaust victims' resources in a similar fashion to the protocol attacks. Contrary to them, layer 7 attacks

target the application layer of the protocol stack. They employ appropriately selected requests that force victims to consume significant resources to respond to them. These attacks are commonly related to web-based applications served over the HTTP/HTTPS protocol. Attackers flood servers with specially crafted requests that (i) are either CPU/memory intensive, e.g. loading multiple files and/or running database queries to return web pages, or (ii) make the server consume its network resources (e.g. sockets exhaustion). Indicative examples for the former category are HTTP GET/POST Floods [52] while for the latter low and slow attacks such as Slowloris [53]. Except for web-based applications, DNS is a common target by application layer attacks. Indicative examples of DNS application layer attacks are DNS Flood/Water Torture attacks [54] that generate random/specially-crafted requests and force victim servers to utilize significant amount of resources to respond. In total, application layer attacks are difficult to be detected in interim network devices, since the attack traffic patterns present similarities to the benign traffic, however these attacks can be pinpointed on victim end-hosts.

Multi-vector attacks refer to simultaneous combinations of the aforementioned attack types. Malicious actors launch multiple attacks against selected victims to (i) bypass network protections schemes, (ii) increase their possibilities to harm the victim, or (iii) hide attack vectors within the attack traffic of other attacks, e.g. launching application layer attacks in parallel with volumetric attacks.

As thoroughly explained, each attack type has its own unique characteristics, however they all share a common goal: to disrupt network services. Integrated DDoS protection frameworks should consider these characteristics and provide fast/accurate attack detection and effective mitigation. In section 4.2, we discuss attacks detection techniques; in section 4.3, we provide details on the countermeasures that can be applied, i.e. mitigation mechanisms.

4.2 Detection Techniques

4.2.1 Overview

As mentioned in the previous subsection, DDoS attacks attempt to impact selected networks/hosts (victims) in various manners, e.g. consume their system resources,

overload the available network bandwidth. The generation of attack traffic creates conditions in computer networks/systems that deviate from the expected/benign state; these conditions can be defined as anomalies. DDoS detection may be defined as the process of identifying such anomalies using as input network data. In general, DDoS detection can be mapped to the following tasks:

- detect the existence of attacks (DDoS Existence)
- identify attack victims (Attack Victim Identification)
- identify the type of the attack (Attack Vector Identification)
- pinpoint malicious flows/packets (Attack Traffic Classification)

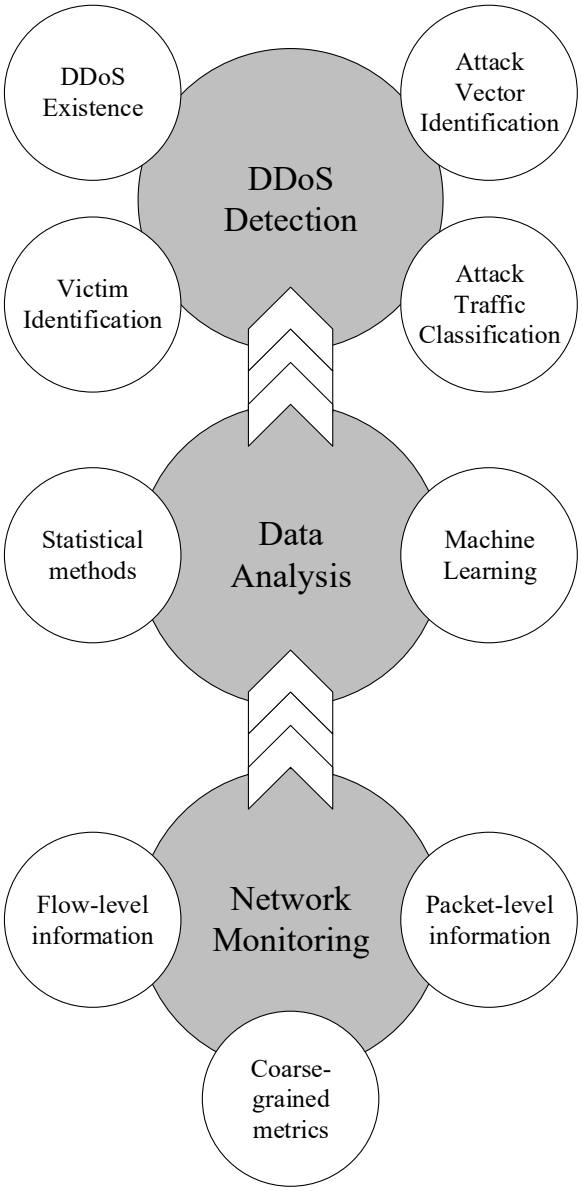

Figure 4.2: General DDoS Detection pipeline

In a nutshell, DDoS detection pipelines gather monitoring data from the network substrate, analyze them, and identify ongoing anomalies. Network monitoring data may include coarse-grained data (e.g. per-port packet/byte counters), flow-level, and packet-level information. These data can be consumed by DDoS detection mechanisms in a streaming fashion and/or periodic time-intervals, depending on the monitoring mechanism (see section 3). The retrieved data are analyzed based on methods that span from simple statistical models (e.g. threshold-based) to more complex algorithms, i.e. *Machine Learning* (*Supervised Learning*, *Unsupervised Learning*). The outcome of this data analysis pertains to the task(s) mentioned above. In Figure 4.2, we illustrate the overall lifecycle of DDoS detection pipelines.

Such pipelines should provide both fast and accurate DDoS detection. Regarding the former, timely detection leads to timely mitigation that can significantly reduce the impact of DDoS attacks. Regarding the latter, detection accuracy is crucial for the benign end-users; benign traffic misclassification results to disallowed benign traffic while malicious traffic misclassification results to attack traffic portions reaching the victim. To that end, high True Positive Rates – TPR (e.g. malicious traffic classified as malicious) and in parallel, high True Negative Rates – TNR (e.g. benign traffic classified as benign) are of paramount importance to minimize the impact of DDoS attacks.

4.2.2 Statistical methods

Most DDoS attacks introduce sudden increases in the incoming traffic rate of the victim network; this is accompanied by abrupt changes in various network traffic metrics, e.g. the number of network flows or even abnormal packet field values. The use of appropriately selected thresholds on network metrics, that during an attack deviate from their expected values, is one of the most common methods for DDoS detection. Basically, this methodology assumes that network traffic characteristics, e.g. the number of flows in a network, follow specific distributions (e.g. *Gaussian distribution*[3]). The values that deviate from the expected behavior are considered anomalies. We present below indicative efforts that rely on this methodology to identify/combat DDoS attacks.

[3] https://en.wikipedia.org/wiki/Normal_distribution

Indicatively, in [55]–[58], threshold-based detection schemes, within SDN (*OpenFlow*-enabled) environments, were proposed; [55]–[57] focus on SYN flood attacks while [58] provides a generic detection scheme. In [55], [57], SYN packets counters are maintained for each IP source and if a predefined threshold value is reached, then the source is considered malicious. These approaches seek to identify malicious sources of TCP floods (Attack Traffic Classification task). In [56], SYN Floods are detected according to abrupt decreases of the entropy value of destination IP addresses. In case the entropy value is lower than a predefined threshold for consecutive time-windows, an active attack (DDoS existence) is assumed, and a victim identification process is initiated. Similarly to [56], in [58] the entropy values for destination IP addresses and ports are calculated; these are compared to preselected thresholds and DDoS attacks existence is indicated. Subsequently, a fine-grained detection scheme is initiated that pinpoints the victim of the attack. This is identified by comparing the number of flows that target each destination IP address to a selected threshold value (excessive values indicate highly asymmetric communication). Finally, thresholds for TCP/UDP packet symmetry ratio are used for malicious sources classification (these are defined according to well-known TCP/UDP traffic patterns).

Threshold-based detection mechanisms have also been reported for programmable data planes. Specifically, in [31], [59] P4-based DDoS detection schemes were proposed. In the former, SYN flood attacks are detected by tracking the per-flow ratio of TCP SYN to regular TCP packets and comparing it to predefined thresholds. In the latter, entropy values of source and destination IP addresses are calculated in the data plane. These values are compared to thresholds and upon their violation a DDoS attack is considered active.

In total, threshold-based methods are a well-established approach for DDoS detection (including all of its tasks) as reported in the literature [31], [55]–[59] but also as validated by tools [60] used in production environments. This approach is commonly preferred due to its interpretability and simplicity; however, it may struggle to follow the continuously evolving DDoS landscape accompanied by complex traffic patterns. Therefore, more sophisticated methods, i.e. *Machine Learning* (ML) algorithms, have raised awareness; these attempt to create generic models for DDoS detection tasks based on multiple features of network data.

4.2.3 Machine Learning

In general, ML approaches are divided into three broad categories, based on the feedback that is returned to a learning system:

- *Supervised Learning*: Example input data (training data) and their desired output values (labels) are provided to an algorithm; this searches for a general rule (function) that maps the given input to the desired output.
- *Unsupervised Learning*: Example input data are provided to an algorithm, which searches for correlations/hidden patterns amongst them.
- *Reinforcement Learning*: The learning system interacts with a dynamic environment and continuously performs actions. These provide rewards to the system, which aims to make the "best" decisions (actions) to maximize a cumulative reward.

Within the context of DDoS detection, algorithms from the aforementioned categories (mainly from *Supervised* and *Unsupervised Learning*) have been widely used for pursuing the detection tasks mentioned above. Specifically, *Supervised Learning* methods use as input labelled network data and classify them to benign/malicious (binary classification[4]) or to attack categories (multiclass classification[5]). *Unsupervised Learning* methods use as input unlabeled network data and attempt to identify hidden correlations by either clustering them into categories or revealing anomalies (outliers). For the latter, the anomaly detection problem[6] is typically transformed to a binary classification problem, in which outliers are considered DDoS attack traffic. Below we present various efforts reported in the literature that employ *Machine Learning* methods for DDoS attacks detection.

In [61], a DDoS detection schema based on a Multilayer Perceptron (MLP) was introduced. Traffic metrics related to flows and packet rates (UDP, ICMP) are collected

[4] https://en.wikipedia.org/wiki/Binary_classification#Statistical_binary_classification

[5] https://en.wikipedia.org/wiki/Multiclass_classification

[6] https://en.wikipedia.org/wiki/Anomaly_detection

and used as input to an MLP, tasked with classifying network traffic to benign/malicious. In [17], *OpenFlow* (OF) entries are collected from network devices, flow-related features are extracted and classified via Self-Organizing Maps (SOM). In [62], an SDN DDoS detection and mitigation schema was proposed. Sharp increases in the rate of OF *Packet-In* messages are considered as an indication of DDoS attacks (threshold-based detection); subsequently *OpenFlow* rules are collected from network devices and classified via an MLP that uses the same feature set as in [17]. In [63], a large set of flow-related features is extracted from packets and sent to OF Controllers. These are used as input to a Stacked Autoencoder, which provides feature reduction and traffic classification of the flow as benign or attack.

In [64] *ATLANTIC*, an SDN framework for DDoS attack detection and mitigation, was proposed. Entropy changes for specific flow features within consecutive time windows indicate the existence of an attack. Network flows responsible for entropy changes are fed in a traffic classification component based on K-means and Support Vector Machines (SVM). K-means is used initially to create clusters of common flows and SVM is further used to identify malicious flows. In [65] *DeepDefense*, a DDoS detection schema based on Recurrent Neural Networks (RNN) was introduced. Traffic traces, collected within sliding time windows, are translated into arrays of packet features. These are fed to an RNN that segregates malicious from benign packets. Similarly, in [66] *LUCID*, network traffic classification also employs packet fields organized in network flows. Packet values are collected from different time windows and organized as arrays; subsequently these arrays are fed to a Convolutional Neural Network to identify time-dependent traffic patterns.

In total, *Machine Learning* methods illustrate high accuracy for DDoS detection tasks, identifying complex attack traffic patterns while also achieving significant generalization capabilities (the ability to detect "unseen" anomalies, i.e. zero-day attacks [67]). In the following subsection, we discuss the main challenges that need to be considered by DDoS detection mechanisms.

4.2.4 Challenges

The key challenges with regards to DDoS detection may be categorized in the following:

- Accuracy:

- Victim Identification: The accurate identification of DDoS attacks victim host/subnet is of vital significance since the instantiation of further protection procedures heavily relies on it. False alarms may lead to delayed identification of the actual victim worsening the impact of the attack.
- Attack Vector Detection: The detection of the exact type of a DDoS attack determines the appropriate type of countermeasures. Falsely reported types can further delay the mitigation of the attack.
- Benign/Malicious Traffic Classification: The accurate identification of benign/malicious traffic is crucial. Misclassified benign traffic affects the quality of experience of legitimate end-users by blocking them from reaching the desired network/service. Respectively, misclassification of malicious traffic allows attacks to flood the victim and consume its resources, downgrading legitimate users' quality of experience (benign and malicious traffic share victims' resources).

- Promptness: Rapid detection of DDoS attacks is of paramount importance, since it enables for immediate enforcement of countermeasures increasing the uptime of targeted networks/services.
- Adaptability: Detection mechanisms should be able to be used in diverse and complex network environments. Methods/algorithms employed for detection tasks need to conform both to the network environment (e.g. ISP) and to the evolution of the network traffic patterns. Robust, reliable, and adaptable detection mechanisms (i) enable for the classification of new (unseen) network traffic patterns and (ii) ease management operations by minimizing operators' manual intervention, i.e. reconfiguration/fine-tuning.
- Scalability: As noted, detection mechanisms consume network data and analyze them to identify attacks. The ever-increasing Internet traffic leads to scalability problems in terms of monitoring data collection/analysis. Thus, such mechanisms should be able with few amounts of network data and within short time-windows to accurately pinpoint anomalies.

Either using statistical methods or *Machine Learning* algorithms to cope with the aforementioned challenges, this is only the first step towards DDoS protection. The next

step is the enforcement of appropriate countermeasures to filter out the attack traffic; this process is defined as DDoS mitigation.

4.3 Mitigation Mechanisms

DDoS mitigation is tightly coupled with DDoS detection. In a nutshell, such schemes retrieve information about the network traffic from DDoS detection mechanisms and enforce filtering rules to block the malicious portion.

We may categorize DDoS mitigation mechanisms based on their deployment location in the following types: (i) *on-premises*, (ii) on *upstream/peer networks*, and (iii) *cloud-based*. *On-premises* approaches mitigate attacks within the network hosting the victim, either using constantly or on-demand protection (e.g. dedicated hardware appliances). This approach is effective for network attacks that do not exceed victim's network links capacity. In that case, one of the (ii), (iii) alternatives need to be considered. Filtering attack traffic on *upstream/peer networks* protects the victim network links, but requires pre-agreements between the victim network and its upstream providers/peers. These may range from typical blackholing [68] to granular filtering [69] techniques. Note that for both cases (i), (ii), the victim network should identify the ongoing anomaly and define the appropriate filtering rules.

An alternative approach, relies on DDoS protection offered by *cloud-based* service providers; these, upon DDoS detection provided by the victim network, drain the network traffic destined to the victim, scrub it, and finally redirect back only the benign portion. *BGP Anycast*[7] is one of the main techniques that enables *cloud-based* scrubbing providers to absorb massive amounts of traffic using dispersed points of presence (POPs) across the globe. Despite its effectiveness, *cloud-based* scrubbing may (i) raise privacy concerns, (ii) introduce additional latency, and (iii) require considerable costs.

The aforementioned approaches can be combined to create hybrid protection schemes, e.g. *on-premises* mitigation for small-scale attacks and *cloud-based* scrubbing for massive attack scenarios; these should be optimized per network environment.

---

[7] https://en.wikipedia.org/wiki/Anycast#Mitigation_of_denial-of-service_attacks

4.3.1 Filtering Methods

We described in the previous subsection various DDoS mitigation services based on their deployment location. Despite this categorization, all of the aforementioned approaches share a common goal, to filter the offending traffic without impacting benign users. In a nutshell, DDoS filtering mechanisms employ one or more packet field values to appropriately distinguish malicious from benign traffic as packets traverse network devices (in the data plane). Matching and filtering capabilities rely heavily on the device type, e.g. router, switch, firewall, COTS server, dedicated hardware appliance. Thus, we present below DDoS filtering techniques emphasizing on their (i) matching capabilities, (ii) drawbacks, and (iii) limitations:

Destination-based Remotely Triggered Black Hole (RTBH) Filtering [70] is a filtering mechanism primarily used to prevent potential collateral damage during DDoS attacks (e.g. bandwidth and CPU utilization, service degradation). It is a destination-based filtering mechanism, in which the traffic destined to the victim is redirected to null interfaces of edge routers and thus dropped. Victim networks use this mechanism for *on-premises* mitigation to protect their network links. However, blackhole filtering is commonly enforced on *upstream networks/peers* to protect victim networks links from congestion. Note that, the main drawback of this mechanism is that both malicious and benign traffic destined to the victim is seamlessly dropped.

Source-based Remotely Triggered Black Hole (RTBH) Filtering [70], unlike the destination-based RTBH, is a source-based alternative that drops packets from specific source IPs using the unicast Reverse Path Forwarding (uRPF) feature [70]. Source-based RTBH also relies on *BGP updates* that contain routes to malicious IP addresses/networks; attack packets from these sources are dropped on uRPF-enabled edge router interfaces. Although more granularity than the destination-based RTBH is offered, packets destined to legitimate destinations may be blocked in *en route* and *fixed route spoofing* scenarios [71].

Access Control Lists (ACLs) is another approach commonly used in switching/routing devices to implement firewall policies. Upon DDoS detection, appropriate ACLs are populated to network devices to block the attack traffic. Contrary to Source-based RTBH, ACLs allow more granular filtering (than source IP addresses). The packet fields that are

commonly used to match and block the offending traffic rely on the 5-tuple of network flows; this is based on the fact that typical detection mechanisms classify network flows to malicious/benign and therefore mitigation mechanisms employ the same set of fields for blocking. ACLs are an effective way for blocking DDoS attacks at the network edge, however they come with some limitations: (i) the number of ACLs is limited in network devices [72], (ii) the supported packet fields that can be used for packet matching are tightly dependent on vendors implementations, and (iii) increased complexity on managing ACLs in multi-vendor environments is introduced.

Similar to ACLs, *OpenFlow* rules support a plethora of matching capabilities and actions/instructions, that may be used for packet rejection [58]. Although OF provides a large number of packet fields that can be used for packet matching, it faces almost the same limitations as the ones reported for the ACLs (except for the complexity of managing them due to the common interface offered by *OpenFlow*).

BGP Flowspec [73] is a filtering mechanism that uses the Network Layer Reachability Information (NLRI) format of *BGP Update* packets to disseminate flow specification rules. These rules extend the capabilities of typical blackhole filtering mechanisms allowing for fine-grained traffic filtering. *BGP Flowspec* rules are one step ahead of typical ACLs, as they provide a 12-field tuple for matching malicious packets while being able to be propagated to network devices over a unified interface, the *BGP* protocol.

Another filtering method for DDoS mitigation is based on packet signatures. These refer to specific packet field values commonly observed in malicious network packets. Signatures have already been classified as malicious and are employed as filtering rules in appropriate middleboxes (DPI is required). These match and block malicious network packets while not affecting benign network traffic. Although this approach is highly effective for well-known attack traffic patterns, it is not able to cope with new "unseen" attacks, i.e. zero-day threats.

4.3.2 Challenges

In a nutshell, DDoS mitigation mechanisms need to:

- support various packet fields, capable to be used for accurate segregation of malicious from benign traffic (in the data plane)

- generate concise and small sets of filtering rules to address data plane memory limitations and simplify/facilitate their management
- account for vertical and horizontal packet processing scalability for elastic on-demand protection
- enable for short filtering rules deployment time for immediate attack alleviation

5 DDoS Detection in Programmable Data Planes (P4)

Data plane programmability is a promising technology that enables rapid control loops for the detection and mitigation of cyber-attacks. In this section, we propose an in-network architecture for DDoS attack detection that combines important traffic metrics of malicious traffic. These pertain to number of flows and packet symmetry, maintained for protected subnets and utilized to identify anomalies. Appropriate alarms are triggered within time-based epochs and conveyed to external mitigation systems. We assess our DDoS detection schema in P4-enabled SmartNICs in terms of detection accuracy and packet processing performance. As input to our accuracy experiments we use real publicly available traffic traces. Furthermore, performance stress tests were conducted using high speed packet generators. Results exhibit that our approach is applicable in typical enterprise and/or carrier environments, featuring packet rates of 1-2 Mpps for 10G links.

5.1 Motivation

As already mentioned, network environments are constantly plagued by massive Distributed Denial-of-Service (DDoS) attacks launched via infected hosts under the control of malicious actors. Accurate and timely DDoS detection is crucial for effective and efficient mitigation. Typical DDoS detection schemas rely on packet samples [74] or flow records [14], exported from agents within network devices (routers, switches). These are relayed for processing to external collectors (servers). Similarly, SDN setups e.g. *OpenFlow* [9] employ control plane signaling between network devices and controllers to trigger detection alarms and subsequent mitigation actions. Such detection mechanisms introduce added overhead on the communication between network devices and external monitoring platforms, thus stalling the attack detection and as a consequence the subsequent mitigation.

In-network DDoS attack detection is a step ahead of legacy detection methods, as it operates directly in the data plane offering rapid attack detection, while enabling control plane triggers to external mitigation systems. To that end, we propose a P4 [75] DDoS detection schema that combines important traffic features to increase accuracy while adhering to performance penalties. In a nutshell, we: (i) inspect network traffic and compute related metrics (features), (ii) evaluate feature values to identify potential threats

and (iii) convey alarms to external systems as digests. These are conducted continuously in short-time intervals enabling timely detection of network anomalies.

The remainder of this section is structured as follows: In section 5.2, we discuss related work; section 5.3 offers an architectural overview and presents the traffic features used; section 5.4 provides implementation details of the proposed solution pertaining to the P4 language; section 5.5 presents experimental evaluations for processing performance and detection accuracy employing benign and malicious (DDoS) traffic traces. Finally, section 5.6 summarizes this section and presents our conclusions.

5.2 Related Work & Contributions

There are various efforts reported in the literature exploring performance capabilities of advanced network applications implemented in programmable data plane environments. In [12], an extensive analysis of the *eXpress Data Path* (XDP) framework is introduced; As mentioned, XDP is a novel approach towards high-performance programmable packet processing in Linux systems. The authors consider use cases such as Routing, DDoS Mitigation and Load Balancing and perform experimental comparisons. In [76], the impact of basic P4 operations (packet parsing, headers modifications) on packet processing performance is explored. Experiments are based on P4-enabled SmartNICs (*Netronome* Agilio CX) and illustrate the effect on processing latency introduced by various P4 constructs. Similarly, in [77], the impact of operations performed within the XDP framework on various system resources is investigated. Specifically, results demonstrate packet processing limitations and scaling capabilities (number of CPU cores) considering different flavors of XDP. Inspired by these approaches, we propose a P4-enabled timely DDoS detection schema and explore its performance capabilities on a SmartNIC-based testbed.

Recent research efforts on data plane programmability applied to detection of DDoS attacks are reported in [31], [59]. In the former, a P4-based DDoS detection approach is proposed; counting *Bloom Filters* are used to track the per-flow ratio of TCP SYN to regular TCP packets in order to detect SYN flood attacks. In the latter, a DDoS detection schema is presented that estimates entropy values of source and destination IP addresses. These values are compared to appropriately defined thresholds and upon their violation a DDoS attack is considered active.

Both approaches employ software switches for experimentation. In contrast we deploy our P4 schema in hardware, i.e. SmartNICs and assess its performance in terms of attainable packet processing rate and detection accuracy. Moreover, [59] focuses entirely on an important attack vector, SYN Floods while [31] detects the occurrence of an attack without indicating the victim. We provide an integrated framework able to promptly detect generic DDoS attacks to specific victim subnetwork, possibly alerting external DDoS mitigation systems via P4 digests.

Note that data plane mechanisms can be employed to enable efficient and programmable filtering (mitigation) based on packet headers [78]. DDoS mitigation is beyond the scope of this section and will be discussed in the next sections.

5.3 DDoS detection in the data plane - High-Level Overview

Our schema is applicable either in transit provider networks (e.g. ISP, Research & Education Network backbones) or customer/edge network domains (e.g. Data Centers, Campus Networks). Upstream network providers may detect network anomalies aiming downstream organizations. Similarly, customer organizations may deploy the same functionality with fine granularity for specific internal subnetworks. Such an indicative architectural setup is presented in Figure 5.1: Traffic originating from various Internet sources is directed towards a P4-enabled edge domain, possibly via a P4-enabled transit provider. We precisely consider the use case of National Research and Education Networks (NRENs) and their Pan-European interconnection GÉANT. NRENs may offer DDoS Protection services to universities and data centers downstream. These services are implemented in P4-capable devices, placed at important vantage points to monitor traffic at different levels of granularity. Specifically, P4 devices: (i) forward network traffic, (ii) maintain important statistics for monitored (sub)networks, (iii) perform anomaly detection tasks and (iv) raise alarms to external mitigation systems.

Our schema maintains a list of specific monitored (sub)networks and/or hosts, depending on the desired granularity level. DDoS attacks are detected by combining the following traffic features: (i) total number of incoming traffic flows (*srcIP*, *dstIP*, *protocol*, *srcPort*, *dstPort*), destined to monitored subnets in a distinct time interval henceforth denoted as "epoch", (ii) significance of a network, characterized by the percentage of flows directed

towards it out of the total incoming flows and (iii) symmetry ratio of incoming to outgoing packets. These features have correlated characteristics and may provide localized alarms for each protected network under generic DDoS attacks.

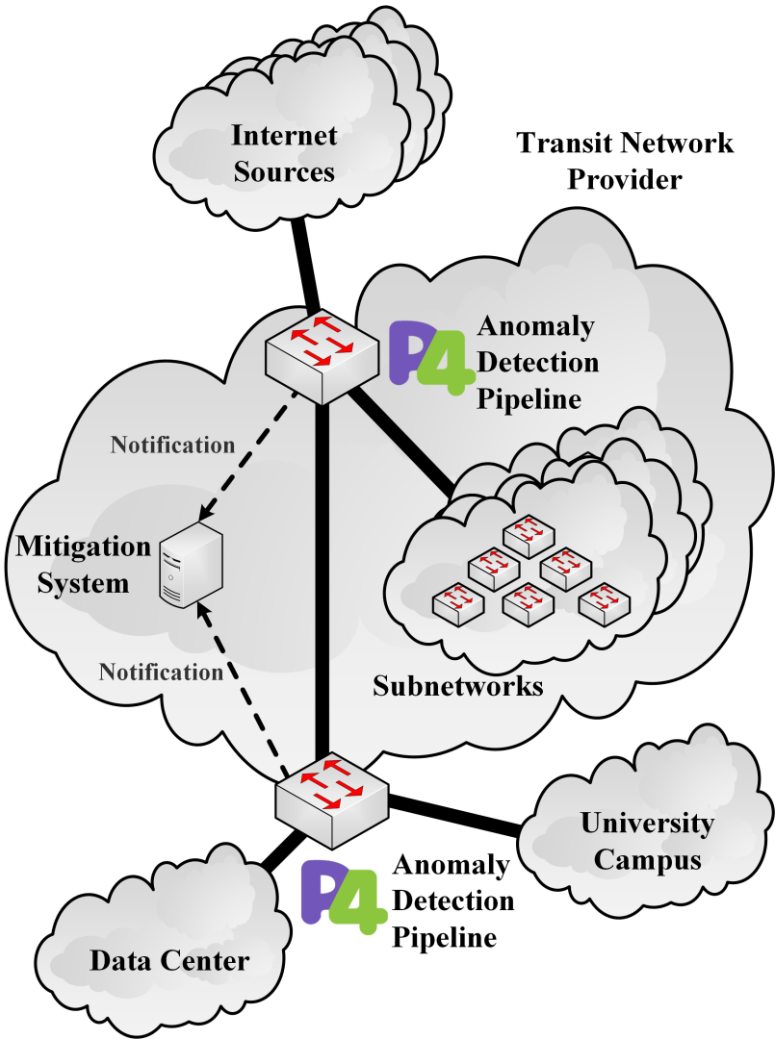

Figure 5.1: High-Level Overview of P4 DDoS Detection Pipeline

Typically, massive DDoS attacks consist of a considerable amount of flows. Thus, we consider the number of total flows as an attack indicator. We adopt a moving average approach as in [31] to track for each epoch n the number and the dispersion of Total Incoming Flows (TIF_n). Specifically, we define M_n as the Exponentially Weighted Moving Average (EWMA) and D_n as the Exponentially Weighted Moving Difference (EWMD):

$$M_n = a \cdot TIF_n + (1 - a) \cdot M_{n-1} \text{ with } M_1 = TIF_1 \quad (5.1)$$

$$D_n = a \cdot | M_n - TIF_n | + (1 - a) \cdot D_{n-1} \text{ with } D_1 = 0 \quad (5.2)$$

The parameter *a* is a smoothing coefficient to dampen short-term fluctuations. Network anomalies are considered if TIF_n exceeds a threshold that depends on M_{n-1} and D_{n-1}:

$$TIF_n > M_{n-1} + k \cdot D_{n-1} \tag{5.3}$$

where $k \geq 0$ is a sensitivity coefficient that scales the detection threshold.

In order to further pinpoint the victim destination subnetwork, we also incorporate two additional features, namely Subnet Significance and Packet Symmetry:

(i) *Subnet Significance* is expressed as the percentage of Incoming Flows $SIF_n^{(i)}$ destined to a subnet *i* in epoch *n* out of the Total Incoming Flows TIF_n. We indicate an alert if this percentage exceeds a significance factor *f* that identifies major flow recipients as potential victims:

$$\frac{SIF_n^{(i)}}{TIF_n} > f \tag{5.4}$$

(ii) *Packet Symmetry* is an insightful metric to avoid classification of a subnet as a victim while it may be a recipient of heavy benign traffic, to which it generates responses. The Current Packet Symmetry Ratio $CR_n^{(i)}$ is defined as the fraction of incoming to outgoing packets for subnet *i* during epoch *n*. These are evaluated based on per subnet *i* counters and compared against a pre-computed Normal Packet Symmetry Ratio $NR^{(i)}$. We consider traffic to a subnetwork anomalous, in case the corresponding fraction exceeds a heuristic threshold *r* as described in the following condition:

$$\frac{CR_n^{(i)}}{NR^{(i)}} > r \tag{5.5}$$

Values for *f*, *r* and $NR^{(i)}$ are defined based on operational experience under normal (non-attack) network conditions. Note that, these parameters could be set by employing *Machine Learning* algorithms that learn from past traffic patterns.

5.4 P4 DDoS Detection Pipeline – Implementation Details

In this section we elaborate on implementation details of the proposed DDoS detection pipeline. Our mechanism utilizes P4 registers to implement counters, arrays and probabilistic data structures. We do not use P4 counters since their values are only

accessible via control plane signaling and may not be used directly in data plane interactions [75]. In Table 5.1, we present indicative register definitions.

Table 5.1: P4 Registers Functionality, Indicative Definition and Usage

Functionality	Example Definition	Usage
Counters	register<bit<32>>(1) epoch	Epochs, Total Flows
Arrays	register<bit<16>>(256) flow_dst	Per Subnet Flows, Packets
Probabilistic Data Structures	register<bit<32>>(65536) sketch	Flow Tracking

In Figure 5.2, we present our processing pipeline. Traffic arriving at the P4-enabled device is filtered to include only relevant packets. Subsequently, we apply our multi-feature approach in distinct serial steps to identify potential attacks. In case all violations are observed, we generate alarms (i.e. P4 digests [75]) to an external mitigation system.

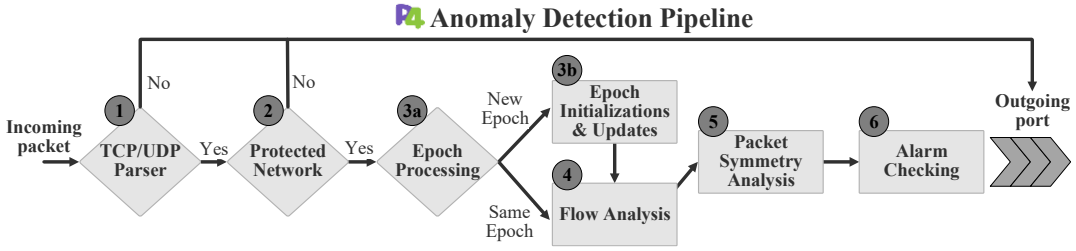

Figure 5.2: Detailed P4 DDoS Detection Pipeline

Step 1 selects only TCP or UDP packets to be considered within the DDoS detection pipeline, since they are typically utilized by most attack vectors. This is achieved using simple checks on parsed headers.

Step 2 further isolates traffic originating from or destined to a monitored network (protected network). To that end, we employ a dedicated match-action table that contains one rule for each protected network. Each rule adds a unique identifier to matching packets as P4 metadata. The added metadata headers are used to access and update the equivalent memory areas of various registers e.g. per subnet measurements such as flows and packet statistics. Note that, traffic that does not meet the aforementioned criteria (i.e. TCP/UDP and source/destination in "monitored" networks), bypasses the DDoS detection pipeline and is appropriately forwarded.

Step 3a delimits time-based epochs, each defined by a start time and duration. Packets are associated with an epoch using the *ingress_global_timestamp* packet metadata. This denotes the exact time a packet arrived at the P4-enabled device. If a packet's timestamp fits within the current time window [start_time, start_time + epoch duration), it is directly fed to Step 4. Otherwise, the packet is assigned to a new epoch and proceeds to Step 3b. The latter performs the following: (i) update the new epoch start time, (ii) increment the index tracking the current epoch, (iii) compute the new EWMA and EWMD values as described in (5.1), (5.2) and (iv) reset the number of total flows.

Step 4 performs flow traffic analysis and maintains appropriate flow counters for packets exiting from either Step 3a or 3b. This operation is based on modified *Bloom Filters* [79], used to track unique active flows within an epoch. Specifically, we calculate hash values from the following packet headers (*srcIP*, *dstIP*, *protocol*, *srcPort*, *dstPort*) that identify a flow tuple. We employ hash functions available in the P4 pipeline, i.e. *CRC32*, *CRC16* and *CSUM16*. The resulting hash values are used as indices to access distinct memory areas of P4 registers. Each area stores the last epoch this flow was observed. A flow is considered "active" in the current epoch when all indices point to register areas containing values equal to the current epoch. Else, the flow is considered as newly observed within this epoch. Subsequently, the register contents for these indices are set to the value of the current epoch. When a new flow is observed, counters pertaining to total flows and per subnet flows are incremented. Based on these counters, conditions pertaining to inequalities (5.3), (5.4) are evaluated. In case a threshold is violated, the equivalent flag is stored in distinct packet metadata headers.

Step 5 performs packet symmetry analysis employing incoming and outgoing packet counters from/to a monitored network. We maintain separate per-subnet packet counters for TCP and UDP traffic, as well as historical normal packet symmetry ratios for both protocols. These are used to evaluate the $CR_n^{(i)}$against the $NR^{(i)}$ as depicted in inequality (5.5). In case this fraction exceeds the threshold r, a flag is raised similarly to the ones for threshold violations (5.3), (5.4).

The final Step 6 of our pipeline checks packets for metadata headers corresponding to identified anomalies. In case all metadata headers are set to "True", an appropriate alarm is generated pinpointing the network under attack and the current epoch. These alarms were implemented as P4 packet digests that enable the communication between the data

plane and external systems; in our case appropriate mitigation mechanisms, able to enforce countermeasures.

Note that, P4 is a programming language with specific restrictions, e.g. no support for floating point arithmetic or division operations. We needed to adapt to P4 limitations using various workarounds since our approach uses real values e.g. the smoothing coefficient *a* in EWMA, EWMD values and divisions e.g. $CR_n^{(i)}/NR^{(i)}$ for its calculations.

Calculations that require floating point operations are approached by multiplying all elements of an equation with a power of 2 and subsequently dividing them by the same factor. Divisions are conducted via appropriate bitwise shifting operations. We present an example for the EWMA equation; specifically, for the smoothing coefficient *a*, we selected the value of 1/256 (~ 0.004):

$$M_n = \frac{1}{256} \cdot TIF_n + \left(1 - \frac{1}{256}\right) \cdot M_{n-1} \Leftrightarrow M_n = (TIF_n + 255 \cdot M_{n-1}) \gg 8 \qquad (5.6)$$

where eight bits right shifting corresponds to division by a factor $2^8 = 256$. We satisfied requirements for division via a plain comparison between two numbers. Note that, we are not interested in the quotient of a fraction but whether it is greater or lower than another value. For example, the threshold evaluation in inequality (3) was implemented as:

$$CR_n^{(i)} > r \cdot NR^{(i)} \qquad (5.7)$$

5.5 Experimental Evaluation

5.5.1 Experimental Setup

In order to validate our DDoS detection framework, we implemented the proposed pipeline in P4 and evaluated it in the testbed illustrated in Figure 5.3. We used as a P4 target the *Netronome Agilio CX SmartNIC* at 10GbE. Programs were developed and compiled via the *Netronome Programmer Studio* while the compiled program was loaded to the NIC. Additionally, we used two VMs operating as the Sender and the Receiver, equipped with 10GbE *Intel* NICs, able to generate and count packets in high packet rates. We assess our DDoS detection schema in terms of detection accuracy and packet processing performance.

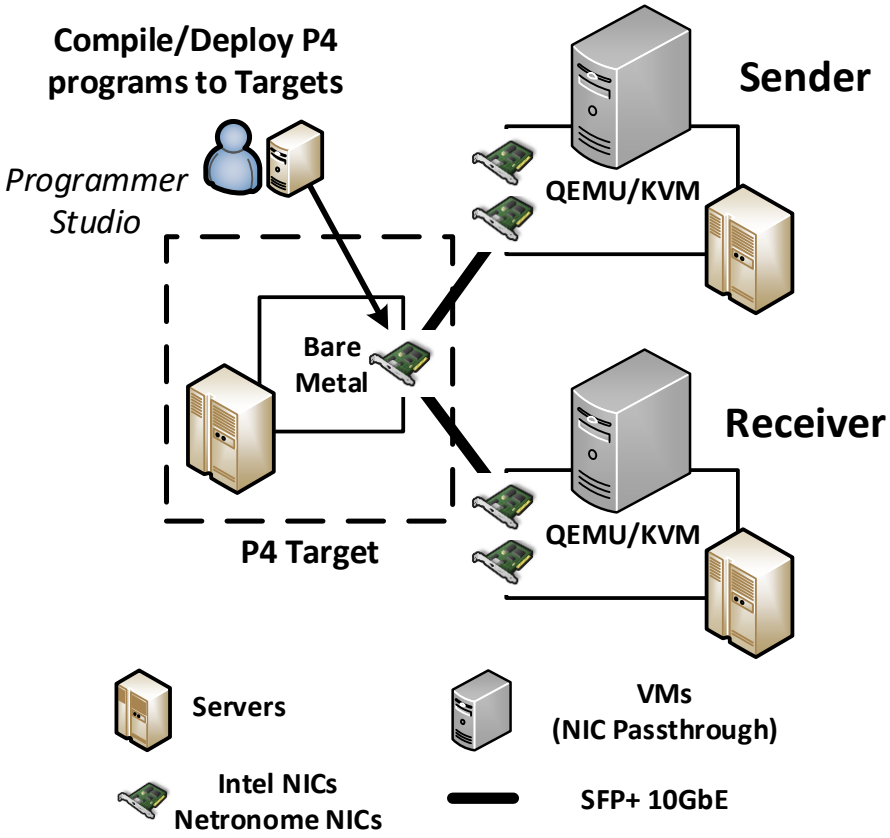

Figure 5.3: Testbed equipped with P4-enabled SmartNICs

5.5.2 DDoS Detection Accuracy

In order to create realistic conditions for our experiments, we used publicly available network traces both for benign and malicious traffic. The benign traffic is based on traces available from the WIDE backbone [80]; specifically traffic from a 10G transit link between WIDE and DIX-IE, an experimental IX in Tokyo. The traces contain network traffic between 12:00 - 12:15 on 09/04/2019. For malicious traffic traces we used the fourth dataset as reported in [2]. This contains a DNS-based reflection attack generated by Booter services. The traces were captured during a controlled experiment conducted by the University of Twente, Netherlands, in collaboration with its upstream provider SURFnet (the Dutch NREN). Protected subnetworks were identified based on an analysis of the benign dataset. We selected the top 255 networks, assuming /24 prefixes, as ordered by the total number of packets traversing each subnetwork.

The experimentation process considered 1-second epochs and was conducted as follows: We injected the benign traffic and ignored alarms for the first 30 seconds, considering

them as a "learning" period for the moving averages. Between seconds 30 and 60 we observed alarms for False Positives. At the 60th second, we launched the attack targeting an IP address within one of the 255 subnets that we monitor. Attack traces were injected between seconds 60 and 90. Packet digests were collected via the *run time API* offered by *Netronome* and used for the calculation of the detection accuracy. Accuracy in binary classification is defined as:

$$ACCURACY = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.8}$$

where *TP*, *TN*, *FP* and *FN* are defined for each subnet in any given 1s epoch:

- *TP*: Number of True Positives i.e. digests received for a subnet when the subnet was the victim of an attack
- *TN*: Number of True Negatives with no digests generated in non-attack cases
- *FP*: Number of False Positives i.e. digests received for a subnet when the subnet was not the victim of any attack
- *FN*: Number of False Negatives with no digests generated in attack cases.

Note that we configured for each subnet only a single digest to be sent during a given epoch.

To showcase the detection capabilities of our mechanism the malicious traces were replayed at different rates. These correspond to three different attack scenarios: (i) an Underscaled attack, i.e. 10% of the reported Booter trace, (ii) the Booter trace as was originally reported and (iii) an Overscaled attack, comprised of 5 times the volume of the reported Booter trace. For all scenarios the benign traffic was injected as it was originally captured.

In the charts of Figure 5.4 we depict accuracy of our framework, evaluated using (4), according to the following empirically inferred values $\alpha = 0.004$, $k = 3$, $f = 0.15$ and $r = 2$, for two cases:

- Two-feature case (F2) that combines conditions (5.3), (5.4) corresponding to Flow Analysis features
- Three-feature case (F3) that also incorporates the Packet Symmetry feature based on condition (5.5)

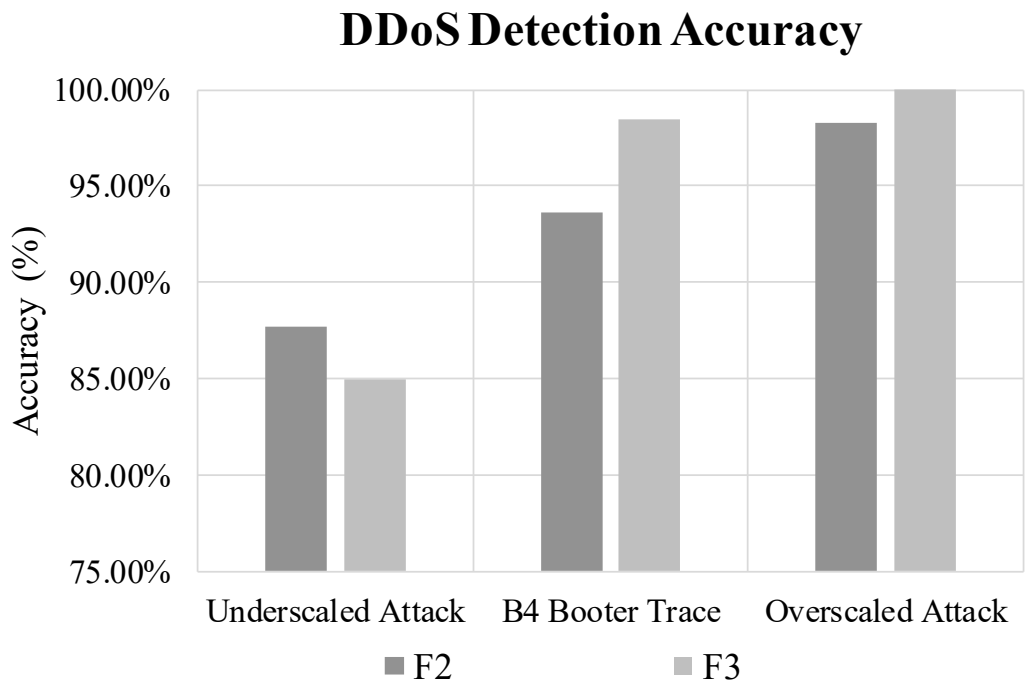

Figure 5.4: P4-based DDoS Detection Accuracy

For the Underscaled attack scenario, F2 performs slightly better than the F3. The former is more sensitive and thus able to identify attacks that generate small fluctuations on the number of flows. The latter, due to the added traffic symmetry feature, misclassifies attack traffic as benign resulting in a considerable number of *FN*s. This occurs since this scenario contains a rather small amount of attack traffic (5% of benign traffic) and packet symmetry ratio does not significantly deviate from the normal (non-attack) values.

For the original Booter trace scenario, both approaches detect the victim, with F3 achieving higher detection accuracy as it has a reduced amount of *FP*s in comparison to F2. Finally, for the Overscaled Attack scenario *FN*s are eliminated due to the vast volume of the attack, achieving accuracy close to 100%. In general, using either two or three features (F2 or F3) we successfully detect ongoing attacks and identify the victim subnetwork within a single epoch.

5.5.3 P4 SmartNIC Packet Processing Performance

We further conducted stress test experiments to assess the processing capabilities of the *Netronome* cards. To that end, we synthesized traffic in various packet rates to (i) assess the performance capabilities of our pipeline and (ii) measure its impact on forwarding throughput. We use the same testbed setup but employ *pf-send* and *pf-receive* utilities of

the *PF_RING* framework [35] on the sender and the receiver respectively. In our experiments we considered the following use cases:

- Plain forwarding case whereby, the target performs only switching (SW)
- One-feature case (F1) that incorporates anomaly identification based on Total Flows evaluation using condition (5.3) only
- Two-feature case (F2) that combines both Flow Analysis features based on conditions (5.3), (5.4)
- Three-feature case (F3) that also incorporates the Packet Symmetry feature based on condition (5.3), (5.4), (5.5)

Note that, the synthesized traffic we used does not bypass our DDoS detection pipeline, thus stressing to the limit the capabilities of the SmartNIC.

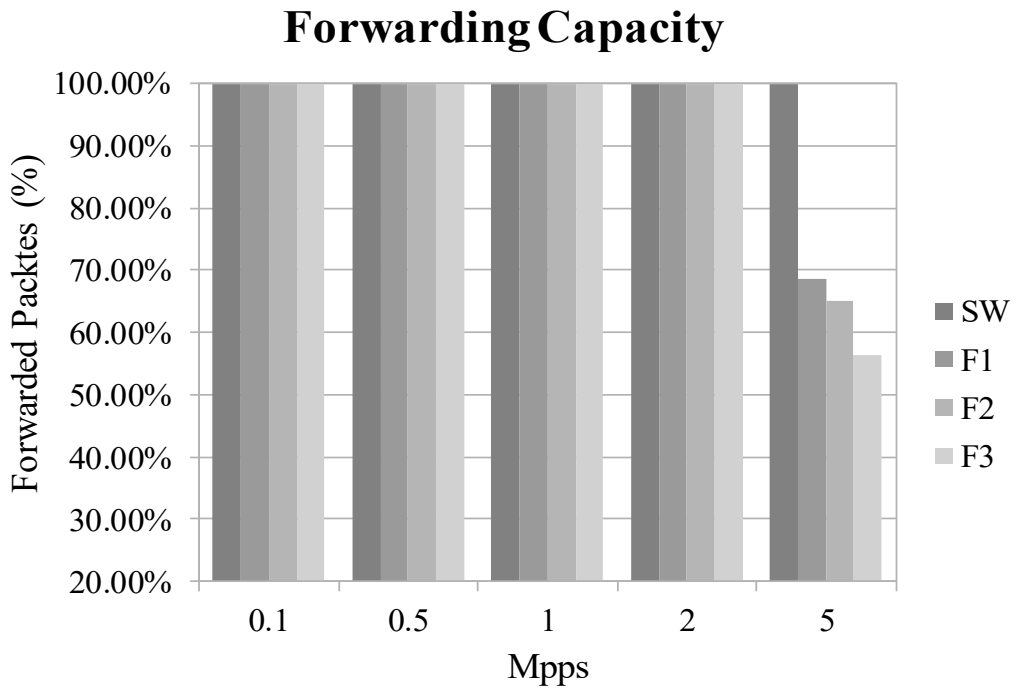

Figure 5.5: *Netronome* SmartNIC Forwarding Capacity

Figure 5.5 depicts the forwarding capacity of *Netronome* cards for various packet rates ranging from 0.1 to 5 Million packets per second (Mpps). The forwarding capacity is calculated as the fraction of traffic that successfully traverses the card.

Traffic rates of 0.1, 0.5, 1 and 2 Mpps show no performance degradation for all four cases. A higher traffic rate of 5 Mpps exhibits considerable degradation of the *Netronome* SmartNIC for adding the DDoS detection pipeline in cases F1, F2 and F3. These amounts

to degradation between 35% to 45%. However, our detection pipeline is relevant in many enterprise and/or carrier networks since 10G links usually correspond to packet rates ranging between 1-2 Mpps according to observations on GRNET network traffic [81].

Our DDoS detection schema heavily depends on accurate packet measurements through SmartNICs. To assess the impact of adding the DDoS detection pipeline, we further investigated the packet counting measurements available in the data plane via P4 registers. These were observed for all cases (SW, F1, F2 and F3) and attainable packet rates (from 0.1 to 5 Mpps), as depicted in Figure 5.6.

For all cases even moderate packet rates of 0.5 Mpps start to exhibit degradation of measurement capabilities. Our DDoS pipeline successfully detects attacks with high accuracy despite measurement limitations of the SmartNICs. As also illustrated in Figure 5.5 our schema does not degrade packet forwarding for rates up to 2 Mpps, typical for 10G links.

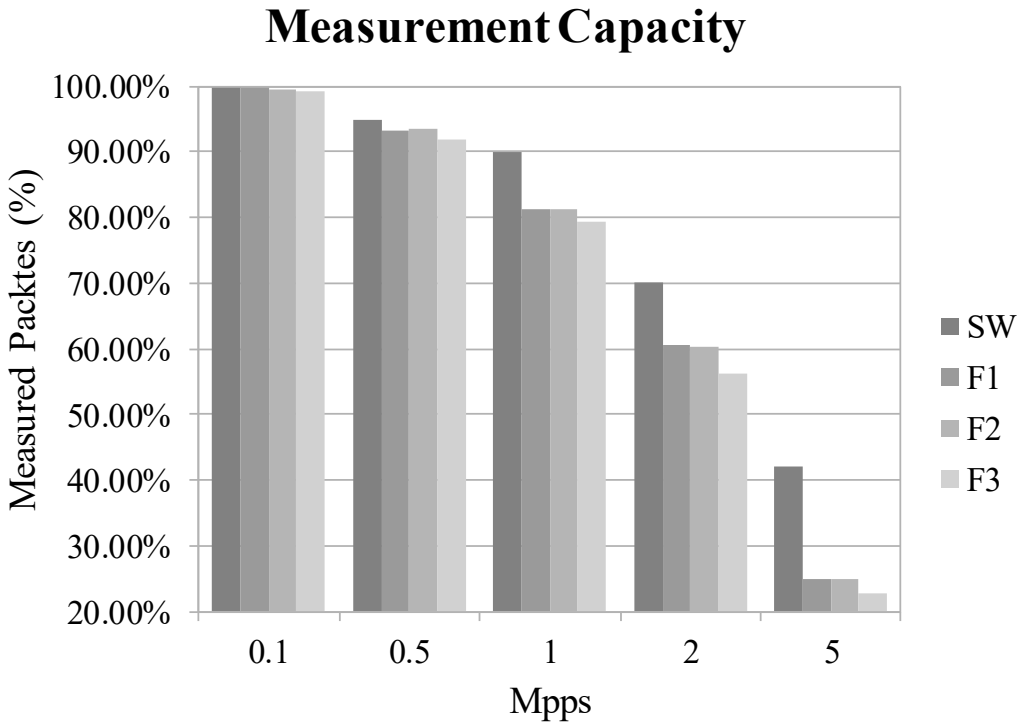

Figure 5.6: *Netronome* SmartNIC Measurement Capacity

These measurement limitations are present only in P4 registers. We have performed additional experiments using P4 counters and observed significant performance improvement. However, as mentioned in Section 4.5, counters are only accessible from external controllers. We attribute this problem to simultaneous accesses of the memory area used for packet counting.

5.6 Summary & Concluding Remarks

In this section we described an in-network DDoS detection schema that combines multiple traffic features. These features are based on typical metrics employed for DDoS detection such as incoming flows and packet symmetry ratio. Our mechanism yields accurate per subnet alarms implemented entirely in the data plane, without any involvement of external controllers, thus enabling rapid control loops. Our experiments for detection accuracy were based in realistic attack scenarios using publicly available traces. We further conducted stress tests using high-rate synthesized traffic to assess the performance of our P4 mechanism, implemented in SmartNICs.

The proposed schema provides an accurate and fast in-network method for detecting DDoS attacks targeting selected victim networks. This can be considered as the first step towards DDoS protection. In the next section we will delve into attack specifics attempting to classify network traffic and filter out the malicious portion. These will be considered for SYN Flood attacks (as an indicative example of protocol-based attacks), a highly employed attack that plagues computer network infrastructures and services.

6 Signature-based Traffic Classification and Mitigation of SYN Flood Attacks using Supervised Learning and Programmable Data Planes

TCP SYN Flood is one of the most common protocol-based DDoS attack that attempts to exhaust memory and processing resources of selected victims. Typical mitigation mechanisms, i.e. SYN cookies consume significant processing resources and generate large rates of backscatter traffic to block them. In this section, we propose a detection and mitigation schema that focuses on generating and optimizing signature-based rules. To that end, network traffic is monitored and appropriate packet-level data are processed to form signatures, i.e. unique combinations of packet field values. These are fed to Supervised Learning models that classify them to malicious/benign. Malicious signatures corresponding to specific destinations identify potential victims. TCP traffic to victims is redirected to high-performance programmable XDP-enabled firewalls that filter offending traffic according to signatures classified as malicious. To enhance mitigation performance malicious signatures are subjected to a reduction process, formulated as a multi-objective optimization problem. Minimization objectives are (i) the number of malicious signatures and (ii) collateral damage on benign traffic. We evaluate our approach in terms of detection accuracy and packet filtering performance employing traces from production environments and high rate attack traffic. We showcase that our approach achieves high detection accuracy, significantly reduces the number of filtering rules and outperforms the SYN cookies mechanism in high-speed traffic scenarios.

6.1 Motivation

SYN Flood (attack) is a major part of the evolving DDoS landscape [82], [83]. This attack exploits the widely employed TCP protocol and especially the 3-way handshake, flooding with SYN packets targeted victims. These exhaust their memory and processing resources, failing to serve legitimate requests. SYN Flood attacks are difficult to counter via commonly used IP-based mitigation schemas. IP-based rules, required to block the attack traffic, increase proportionally to the number of malicious sources. This demands network devices/firewalls to store thousands/millions of filtering rules, which is unattainable due to memory resources limitations [72]. Notably, when spoofing is employed, IP-based filtering is totally ineffective. An alternative mitigation method for

SYN Floods, relies on the *SYN cookies* [84] technique. This approach, instead of blocking malicious SYN packets, generates appropriately crafted SYN-ACK packets. Although, this method protects the victim from the launched attack, it consumes significant processing resources and introduces large rates of backscatter traffic [85].

Inspired by the aforementioned challenges, we propose a signature-based mechanism for SYN Floods detection and mitigation. Our mechanism collects network data and extracts appropriate packet fields, forming packet signatures. Subsequently, these signatures are used as input to *Supervised Learning* models tasked with classifying them to malicious/benign. Malicious signatures corresponding to specific destinations identify potential victims. TCP traffic to victims is redirected to high-performance programmable XDP-enabled firewalls that filter offending traffic according to signatures classified as malicious. To enhance mitigation performance malicious signatures are subjected to a reduction process, formulated as a multi-objective optimization problem.

The remainder of this section is structured as follows: In Section 6.2 we discuss background information and related work; Section 6.3 presents a high-level overview of the proposed mechanism and its core design principles; Section 6.4 provides implementation details for the SYN Flood detection and mitigation architecture; Section 6.5 presents experimental evaluations for detection accuracy and packet filtering performance using both benign and malicious traffic captured in real network environments. Finally, Section 6.6 summarizes our work and discusses further extensions.

6.2 Related Work & Contributions

There are many efforts reported in the literature related to SYN Flood mitigation. In [55]–[57], SDN controllers act as proxies protecting servers targeted by SYN Flood attacks. Specifically, they respond to each received benign or malicious SYN packet with a SYN-ACK packet. Legitimate ACK responses are correlated with previously observed SYN packets and henceforth validated clients can initiate TCP connections. In such approaches SDN controllers store SYN packet monitoring data (e.g. source IP, destination IP, source port, destination port) that may lead to memory exhaustion; added latency is also introduced due to network traffic interception by the controller.

An alternative method for mitigating SYN Flood attacks is based on the *SYN cookies* technique [49], [84], [86]. In this approach, for each SYN a SYN-ACK response is

generated using as sequence number a specially crafted value (cookie). This value is calculated based on hashing operations on IP and TCP packet fields of the received packet, combined with timestamp values. Subsequently, legitimate clients send an ACK as a response to the SYN-ACK setting the acknowledgement number equal to the cookie (sequence number) value increased by one. The acknowledgement number of the ACK is compared to the cookie value, calculated based on the IP and TCP header fields of the ACK. If these values are equal, the client is considered legitimate and henceforth connections from this client are accepted, else the ACK is dropped. Instead of consuming memory resources to store details related to the client, this approach saves information in the sequence number of the SYN-ACK packet via the *SYN cookies* mechanism. Notably, in [87] the *SYN cookies* mechanism was implemented in P4 and tested in various hardware targets, e.g. NetFPGA, SmartNICs; such approaches achieved remarkable SYN Flood mitigation performance.

Despite *SYN cookies* mitigation effectiveness, there are two major drawbacks: it (i) wastes significant packet processing resources for *SYN cookies* calculation to respond to malicious SYN packets and (ii) floods Internet with SYN-ACK responses equal in rate to the malicious SYN packets. The latter may lead to further network congestion.

Considering the aforementioned drawbacks and inspired by Cloudflare's mitigation approach [22], [85], we propose a signature-based detection and mitigation mechanism for SYN Flood attacks, where:

- Unique combinations of packet field values, i.e. signatures, are dynamically classified based on *Supervised Learning* algorithms; signatures are used to identify ongoing attacks.
- Malicious signatures are used as filtering rules for mitigation purposes. Mitigation performance is enhanced via a rule reduction process formulated as a multi-objective optimization problem.
- The reduced set of filtering rules is deployed on high performance programmable firewalls (XDP) to efficiently mitigate SYN Flood attacks.

6.3 High-Level Overview & Design Principles

We present a high-level overview of the proposed architecture for detection and mitigation of SYN Flood attacks. Our mechanism consists of the (i) Signature Classification, (ii) Signature Reduction and (iii) Anomaly Mitigation components.

As illustrated in Figure 6.1, traffic monitoring mechanisms continuously extract packets (Monitoring Data) from the border (edge) router. These are aggregated, within configurable time windows based on distinct signatures, i.e. IP and TCP header values. In the Signature Classification component, packet signatures are fed to supervised ML models, that classify them as benign or malicious. The classified signatures are used to (i) detect ongoing SYN Flood attacks, (ii) identify the victim and redirect corresponding TCP traffic and (iii) create appropriate filtering rules to mitigate the anomaly.

Meanwhile, TCP packets destined to the victim, are redirected at the border router via the Attack Redirection mechanism to the Mitigation component. To improve the mitigation performance, malicious signatures are subjected to a reduction process (Signature Reduction) before being used as filtering rules. This reduction is formulated as an optimization problem, in which combinations of packet features are explored, that minimize simultaneously (i) the number of signatures required to block the attack traffic and (ii) collateral damage on the benign traffic.

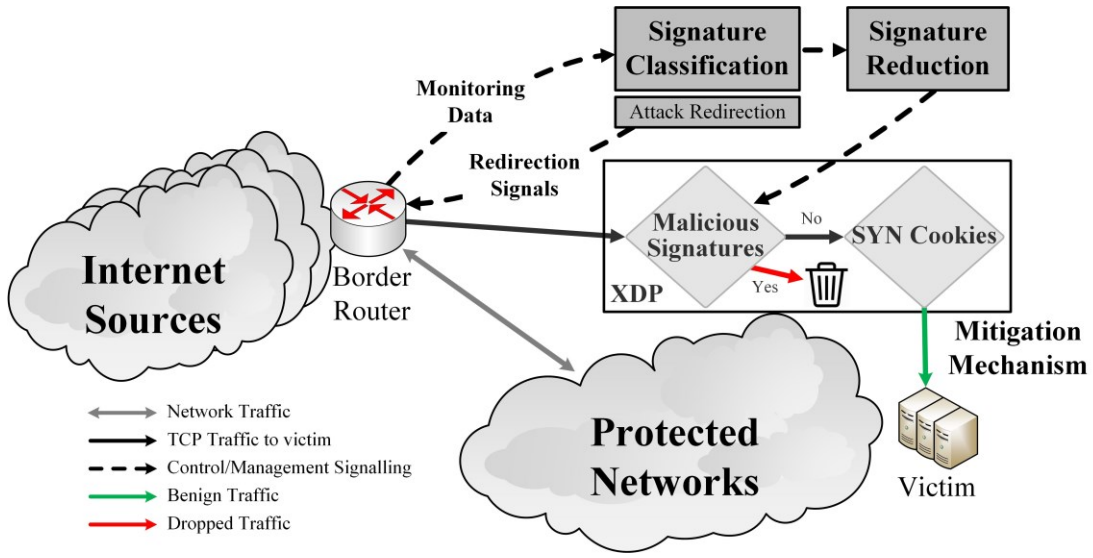

Figure 6.1: SYN Flood Detection and Mitigation Architecture

Initially, the Mitigation component employs temporarily the *SYN Cookies* technique to prevent malicious traffic to reach the victim. When signature reduction is completed, the reduced set of signatures is installed to the Mitigation component. These rules match and drop the offending traffic while benign traffic is appropriately forwarded to the victim.
We present below the core design principles of the proposed architecture:
Signature-based filtering: Our approach identifies malicious signatures and redirects TCP traffic destined to the victim for fine-grained filtering enabled by software data planes (XDP). In contrast to this approach, commonly used mitigation mechanisms rely on the *SYN Cookies* technique that employs significant processing resources to respond to malicious SYN packets and generates backscatter traffic.

Supervised Machine Learning traffic classification: Signature classification is conducted based on *Supervised Learning* models trained a priori with benign and malicious traffic. This enables SYN Flood detection and malicious signature identification based on previously observed benign and attack traffic patterns.

Signature reduction: We introduce a signature reduction mechanism that identifies the signatures required to fully block an attack, minimizing simultaneously their number and collateral damage on benign traffic. This approach attempts to reduce the number of filtering rules. These are stored within network devices that typically impose limits to the number of rules they can support.

High-performance programmable firewalls: We leverage capabilities offered by softwarized programmable data planes (XDP) to design and implement high-performance firewalls. These can be tuned and optimized based on the identified malicious signatures to block SYN Floods in an efficient and flexible manner.

6.4 SYN Flood Detection and Mitigation Architecture

The proposed schema for SYN Flood detection and mitigation consists of the following components: (i) Signature Classification, (ii) Signature Reduction and (iii) Mitigation Mechanism. These are described in detail in the following subsections.

6.4.1 Signature Classification

This component receives and analyzes TCP packet-based data from external monitoring mechanisms to identify malicious signatures. Packet-based data extraction may be conducted either via (i) dedicated XDP mechanisms deployed in-line across the forwarding path, (ii) passive methods such as monitor ports and optical taps, or (iii) data export protocols such as *sFlow* [74]. Note that (i), (ii) may account for all network traffic while (iii) is based on packet sampling. The exact implementation of monitoring data extraction is not the main focus of this work; our only requirement is access to L3-L4 packet headers.

Extracted data are aggregated within configurable time windows, based on appropriate packet fields (features) forming signatures. Our scheme focuses on the relevant features for SYN Flood traffic classification. Specifically, we removed features that have the same value for both benign and malicious packets (zero variance). From the remaining features, we also excluded IP and TCP header length and checksum fields as irrelevant to the classification process. Finally, we excluded TCP sequence number as this value is randomly generated by each client. The final set of features that we employed are:

Table 6.1: Packet fields (features) for TCP SYN packet classification

Packet Field	Short Description
ip.src	Source IP Address
ip.dst	Destination IP Address
ip.dsfield.ecn	Network Congestion Notification
ip.id	IP Fragment Identification
ip.flags.df	Do not Fragment Bit
ip.ttl	Time To Live
tcp.srcport	Source Port of TCP Segment
tcp.dstport	Destination Port of TCP Segment
tcp.window_size	TCP Receive Window Size

Although these features correspond to numerical values, we employed them as categorical variables considering that their actual values are not relevant to TCP SYN traffic classification. These categorical data should be encoded before being used in ML methods. To that end, we used an encoding scheme that for each signature, calculates the frequency of each feature. In the following Table we illustrate an example of the employed frequency encoding scheme considering 5 packets and 2 features (ip.ttl, ip.dst).

Table 6.2: Frequency encoding for categorical features

ip.ttl	ip.dst	#Packets	ip.ttl_freq	ip.dst_freq
239	192.168.1.1	3	60%	80%
62	192.168.1.1	1	20%	80%
61	10.1.1.1	1	20%	20%

Frequencies are calculated as the number of times a packet field value (e.g. ip.ttl 239, ip.dst 192.168.1.1) is observed in a time window, divided by the total number of packets observed in the same time window. This may reveal packet field values that are abnormally frequent during ongoing attacks (or the opposite). The frequency encoded features (e.g. ip.ttl_freq, ip.dst_freq) are used as input to supervised ML models (Random Forest - RF or Multilayer Perceptron - MLP), that classify them as benign or malicious. Signatures are in turn labeled based on the classification of their corresponding frequency encoded features.

If a single signature is classified as malicious, the Signature Classification component notifies the Attack Redirection mechanism. This triggers the border router to redirect traffic destined to the victim (information obtained from the destination IP) to the Mitigation component. Simultaneously, benign and malicious signatures are processed to generate the appropriate number of filtering rules for attack mitigation.

Note that traffic may be redirected either using Policy Based Routing or *BGP* (*Flowspec*) [69]. Further implementation details are beyond the scope of this work.

6.4.2 Signature Reduction

Signatures classified as malicious, are going to be employed as filtering rules for attack mitigation purposes. These rules are stored in memory resources, i.e. *BPF Maps* that enable packet matching at line-rate. Their number significantly affects the deployment and lookup time in the *BPF Map*, which in turn degrades the overall mitigation performance. Therefore, the Signature Reduction component is tasked with providing a concise set of signatures (filtering rules) that can block all the offending traffic, without significantly affecting the benign traffic.

We formulated this signature reduction as a multi-objective optimization (feature selection) problem, in which we search for feature combinations $F' = \{F_1, F_2, \dots F_j\}$, subsets of the initial feature set $F = \{F_1, F_2, \dots F_n\}$, where $j < n$, that simultaneously minimize:

i. the number of malicious signatures (filtering rules) that block all the attack traffic (Count-distinct problem[8])
ii. the percentage of benign traffic that is dropped

We define as M and B the sets of malicious and benign signatures respectively, based on features from F (see Table 6.1). For each combination F', we calculate M' and B', that correspond to sets M and B using only the features of F'. The first objective (i) is calculated as the number of unique signatures (cardinality) in M'. For the second objective (ii), we calculate the number of benign packets that correspond to the signatures in $M' \cap B'$ and divide it with the number of benign packets that correspond to the signatures in B'. This provides the percentage of benign traffic that would be dropped if we used as filtering rules the signatures in M'. Note that the intersection $M \cap B$ is an empty set, however, the intersection $M' \cap B'$ may result to non-empty sets in the reduced feature space F'.

The proposed optimization problem leads to multiple Pareto optimal solutions. However, due to stringent time constraints for attack mitigation (DDoS attacks should be blocked

[8] https://en.wikipedia.org/wiki/Count-distinct_problem

as early as possible), brute-force algorithms may not be able to identify optimal solutions. We opted for a fast evolutionary approach based on Non-dominated Sorting Genetic Algorithm II (NSGA-II) [88]. The algorithm starts with arbitrary subsets $F' \subset F$, and iteratively attempts to enhance the solutions quality, i.e. minimize further the objectives. At each iteration (generation), new subsets of F are generated based on random combinations of F' that correspond to the best solutions produced in the previous iteration. The algorithm stops when a time limit is reached thus generating suboptimal subsets.

As mentioned, more than one solution may be generated but only one of them can be selected for blocking the attack traffic. This selection should be tuned per network environment to depict network operator preferences e.g. acceptable percentage of dropped benign traffic. Finally, from the selected solution, signatures of M' are conveyed to the Mitigation component to be applied as filtering rules.

6.4.3 Anomaly Mitigation

This XDP-based component inspects TCP traffic and prevents malicious TCP SYN packets to reach the victim. As a first level of protection, it filters malicious SYN packets based on the signatures emerged from the reduction process. Packets not filtered at this level, are processed and handled appropriately by the *SYN cookies* mechanism.

The Anomaly Mitigation component parses and isolates TCP packets. Subsequently, it extracts from TCP SYN packets appropriate TCP/IP packet fields which are compared to the signatures stored in a *BPF Map* (hash table). If the extracted signature exists in the *BPF Map*, the packet is considered malicious and is dropped, else it is conveyed to the *SYN cookies* mechanism. This is used: (i) as an initial countermeasure upon the detection of SYN attacks, until signature reduction is completed, (ii) as a fallback mechanism to our signature-based approach for malicious traffic falsely classified as benign and (iii) to validate and allow benign TCP traffic to be forwarded to the victim.

The *SYN cookies* mechanism was implemented within the XDP framework according to the description provided in section 6.2. Further implementation details for the Mitigation component are available in our code repository [89].

6.5 Experimental Evaluation

In order to evaluate our framework, we implemented all software components of the proposed architecture and deployed them in our laboratory testbed. *Supervised Learning* models of the Signature Classification component were based on the *sklearn* and *pytorch* python libraries. The Signature Reduction mechanism was based on the *Platypus* framework [90], used for solving our multi-objective optimization problem. The Mitigation Mechanism was deployed on a physical machine equipped with a 10G XDP-enabled SmartNIC *Netronome Agilio CX*. This was directly connected to a Virtual Machine employed as a high-speed packet generator based on the *PF_RING ZC* framework [35].

We assessed our mechanism detection accuracy and packet filtering performance using both benign and malicious network traces. In subsection 6.5.1, we provide details for the datasets we used. In subsection 6.5.2, we compare the detection accuracy of two commonly used *Supervised Learning* methods (Random Forest, Multilayer Perceptron). In subsection 6.5.3, we showcase our signature reduction mechanism and in subsection 6.5.4, we compare the performance of our approach to the *SYN cookies* mechanism.

6.5.1 Datasets Description

As benign traffic, we used traffic traces from a 1G transit link between WIDE and an upstream provider [80]. We isolated TCP SYN packets captured at 12:15 and 12:29 on 08/04/2020; these are respectively referred to as B_1 and B_2 for the remainder of this subsection. As malicious traffic, we used 5 different TCP SYN attacks that targeted our infrastructure (May – September 2020). The characteristics mentioned in Table 6.1 for the five attack datasets (i.e. A_1, A_2, A_3, A_4 and A_5) as well as their packet rate (Kilo packets per second – Kpps), are presented in Table 6.3 and Figure 6.2 below:

Table 6.3: Packet Feature Cardinality for A1-A5 SYN Flood attacks

Attack	ip.src	tcp.srcport	tcp.dstport	ip.id	ip.ttl
A_1	15	65535	65535	1	3
A_2	760863	65534	65534	1	4
A_3	839660	65535	65535	1	4
A_4	3415575	65536	1	65535	2
A_5	1493948	65536	1	65535	3

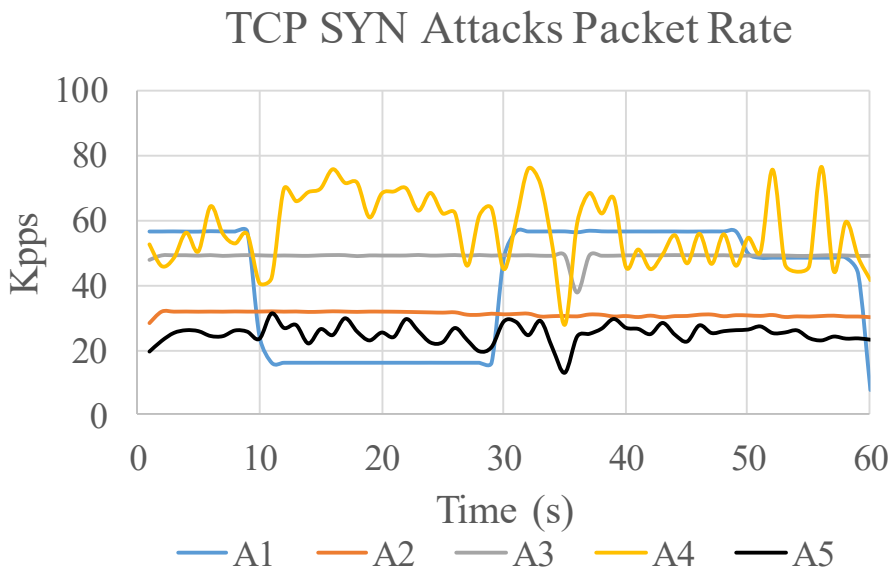

Figure 6.2: Packet Rate of SYN Flood Attacks

As illustrated in the Table above, all attacks except A_1 emanate from a vast amount of unique IP sources. A_1, A_2 and A_3 are using all available source and destination ports and have a single value related to the *ip.id* (IP fragment identification). In contrast, A_4 and A_5 are using a single destination port and have multiple values related to the *ip.id*. All attacks have a small number of unique *ip.ttl* values and a single value for each of the following packet fields: *ip.dst*, *ip.dsfield.ecn*, *ip.flags.df* and *tcp.window_size*.

6.5.2 Signature Classification Accuracy

In this subsection, we evaluate the detection accuracy of the proposed Signature Classification mechanism using two different *Supervised Learning* alternatives: (i) Random Forests (RF) consisting of 100 decision trees with default parameters of the *sklearn* library for tree structure and stopping [91] and (ii) Multilayer Perceptron (MLP) of 9 input neurons corresponding to the selected features, 19 hidden and a single output node for classification. The MLP was trained using sigmoid as the activation function, early stopping for the number of epochs, batches equal to 2048 and Adam method [92] for weight updates with learning rate $\alpha = 0.01$. The validation dataset was set equal to the 30% of the training dataset.

We considered various training scenarios (in total 30) using each unique combination of benign and malicious traffic datasets for:

- Each method (RF, MLP)
- Each attack dataset A_i, where i = 1...5
- The benign dataset B_1
- Each time window (5s, 10s, 30s)

Resulting trained models were used to test the accuracy of our schema for the same time window. The test datasets consist of the benign dataset B_2 not used for training and the attack datasets A_i. This correspond to a total of 5 testing scenarios.

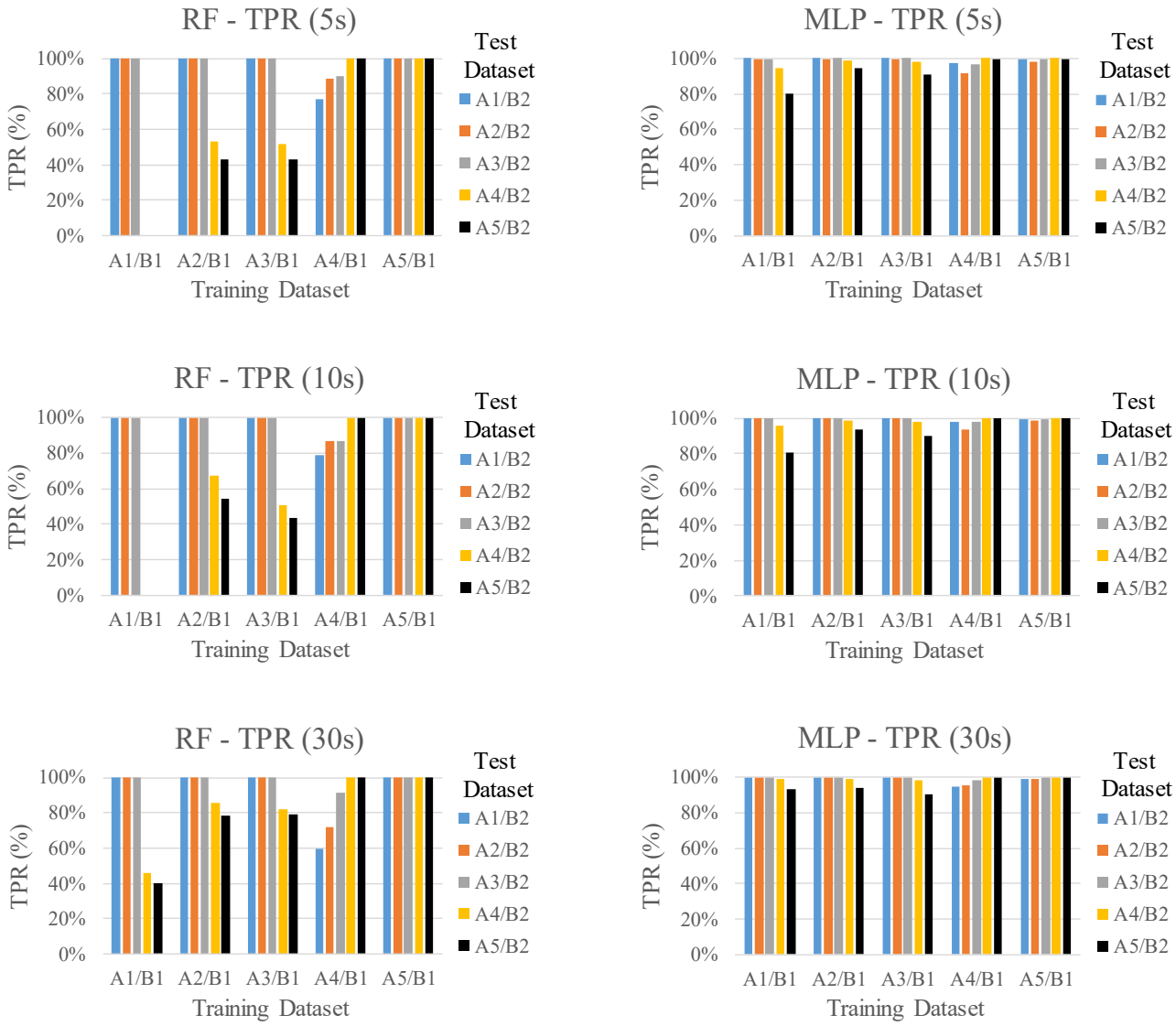

Figure 6.3: True Positive Rate for training/testing scenarios combining benign and malicious TCP SYN traffic

In Figure 6.3, we present results for the aforementioned training and testing scenarios. Specifically, we illustrate the True Positive Rate - TPR, which is the percentage of the attack traffic that was classified as malicious. We do not present results for the True

Negative Rate - TNR (the percentage of benign traffic classified as benign), as the TNR for both ML methods and all time windows was above 99%. The results presented in Figure 6.3 correspond to average values of 100 training-testing experiments to account for random variations in our supervised ML methods. This number was selected based on our experience as a reasonable choice to dampen statistical outlier values.

As illustrated in Figure 6.3, when a model is trained with one of $\{A_1, A_2, A_3\}$ attack datasets, it can accurately detect attacks in the same group, regardless of the time window. This also happens for the group of $\{A_4, A_5\}$ since attacks of the same group have common characteristics as mentioned earlier (see Table 6.3). Moreover, when both classification algorithms (RF, MLP) are trained with A_5, they can accurately identify all attacks. This is attributed to A_5 characteristics; namely, low packet rate (compared to the other attacks), that enables both algorithms to identify greater frequencies of packet features as attacks.

In general, RF is a consistent method to identify TCP SYN attacks, provided the training dataset is similar to the test dataset (e.g. training dataset A_2 - test dataset A_1 or A_3). The accuracy of this model increases as the duration of the time window increases; however, it does not always detect attacks that deviate from the training dataset (e.g. training dataset A_1 - test dataset A_4 or A_5). In contrast, the MLP model identifies for every training scenario all other attacks with high accuracy, illustrating significant generalization capabilities. Notably, it achieves remarkable accuracy even within shorter time windows, e.g. 5s.

In total, our signature classification mechanism achieves high TPR identifying almost all malicious signatures without significant benign traffic misclassification (this was lower than 1% in all our experiments).

6.5.3 Signature Reduction Evaluation

In this subsection, we evaluate our signature reduction mechanism based on the solutions generated by *NSGA-II*. All signatures were extracted from each attack dataset A_i and benign dataset B_1. Each combination (A_i+B_1) was used as an input to the proposed signature reduction mechanism. In Table 6.4, we present for each dataset combination all solutions that resulted from 100 different executions of the NSGA-II. The maximum processing time for each execution was limited to 20s.

For each solution in Table 6.4, we illustrate the percentage of dropped benign traffic (%), the number of reduced signatures and the total signature reduction (%) based on the total number of signatures of each attack A_i. Note that bold values in the following Table correspond to Pareto-optimal solutions.

Table 6.4: Signature Reduction solutions provided by NSGA-II

Datasets	Benign Traffic Dropped (%)	Total Signatures	Signatures (reduced)	Signature Reduction (%)
$A_1 + B_1$	**0**	758078	**1**	99.999
	0.0002		1	
	0		3	
	0.0006		3	
	0		15	
$A_2 + B_1$	**0**	1070311	**1**	99.999
	0.0006		1	
	0		4	
$A_3 + B_1$	**0**	1331799	**1**	99.999
	0.0006		1	
	0		4	
	0.0002		4	
$A_4 + B_1$	**0.151**	3417663	**1**	99.999
	0.17		1	
	1.398		1	
	1.401		1	
	2.123		1	
	2.126		1	
	0.031		**2**	
	0.177		2	
	0.229		2	
	0.643		2	
$A_5 + B_1$	**0**	1494425	**1**	99.999
	0.001		1	
	0		3	

The results Table 6.4 demonstrate that our mechanism significantly reduced the number of signatures (ranging from hundreds of thousands to millions) to a small set of filtering rules (99.99% reduction). The generated solutions provide signatures that can fully block the offending traffic, without affecting significant portions of the benign traffic.

Note that our experiments demonstrated a dramatic signature reduction. This is due to specific packet field values of the attacks we captured and may vary under different attack scenarios. Similar observations have been reported by Cloudflare in [93].

6.5.4 SYN Flood Mitigation Performance

In this subsection, we assess the packet filtering performance of the proposed Mitigation mechanism using two variants: (i) *BPF MAP* and (ii) *STATIC*. The former employs a *BPF Map* for storing signatures while the latter constructs signatures via appropriate static if-else statements [22]. These were compared in terms of packet filtering performance to (iii) the *SYN Cookies* approach, that we also implemented in XDP.

Synthesized traffic was generated based on attack dataset A_1 and replayed at 10 Million packets per second (Mpps). We compare the performance of each approach (i), (ii), and (iii) based on the percentage of traffic that they can successfully drop. Note that we used a single core for packet filtering in our experiments. For the two variants of our mechanism, we employed 3 signatures (see third row of Table 6.4).

Table 6.5: SYN Flood mitigation performance

XDP Implementation	Packets blocked (%) out of 10Mpps
BPF MAP	70%
STATIC	92%
SYN cookies	47%

Both variants of our approach achieve greater packet filtering performance than the *SYN cookies* (from 47% to 70% and 92%). This is attributed to the complex operations that are required to be done for each SYN packet in the *SYN cookies* technique, i.e. cookie calculation, IP/TCP checksums. In contrast, our signature-based approach matches specific packet fields and drops the offending traffic. In case signatures are constructed via appropriate if-else statements, memory lookups are fully avoided and thus packet filtering performance increases even further. Note that, our approach apart from significantly outperforming the *SYN Cookies* mechanism, does not generate backscatter traffic that may introduce further congestion.

6.6 Summary & Concluding Remarks

In this section we proposed a signature-based detection and mitigation schema for SYN Flood attacks. Our schema collects and analyzes, within time windows, appropriate packet data forming signatures. These are subsequently used as input to supervised *Machine Learning* models that detect SYN attacks, identify victims, and isolate malicious signatures. TCP traffic to the victim is redirected to high-performance programmable XDP-enabled firewalls that mitigate identified attacks. Malicious signatures are employed to block the offending traffic, after being subjected to a reduction process to enhance mitigation performance. Signature reduction was formulated as a multi-objective optimization problem that attempts to simultaneously minimize the number of filtering rules and collateral damage on benign traffic.

Our approach was evaluated both in terms of detection accuracy and packet filtering performance. The conducted experiments illustrated high detection accuracy for real benign and malicious traffic. Notably, our mechanism dramatically reduced the number of signatures (filtering rules) required to block the considered attack datasets. Moreover, our approach outperformed the state-of-the-art SYN Flood mitigation mechanism, i.e. *SYN cookies*.

In the next section, we extend the work proposed in this section to volumetric attacks. Moreover, we consider an automated way for selecting packet fields for signature formulation, minimizing the human intervention in the initial feature selection. Finally, we perform a thorough comparison between signature-based and the state-of-the-art flow-based classification/filtering mechanisms.

7 Signature-Based Traffic Classification and Mitigation: Volumetric DDoS Attacks

Distributed Denial-of-Service (DDoS) attacks mitigation typically relies on source IP-based filtering rules; these may present scaling issues due to the vast number of involved sources. In this section, we propose a source IP-agnostic DDoS traffic classification and filtering schema for volumetric attacks that identifies malicious packet signatures via supervised Machine Learning methods and subsequently generates signature-based filtering rules. To accelerate packet processing, our schema utilizes XDP middleboxes operating as programmable Deep Packet Inspectors. Signatures are extracted from network traffic as unique combinations of the most significant packet features; these are subsequently fed to supervised Machine Learning algorithms that classify them as malicious or benign. Malicious signatures undergo a reduction process tailored to the attack vector to generate a concise set of filtering rules, thus expediting mitigation performance. Our schema was implemented as a proof-of-concept and evaluated for DNS volumetric attacks in terms of signature classification accuracy and packet filtering throughput. Experiments were based on benign and malicious traffic datasets recorded in production network environments. Our approach was compared to source-based mechanisms in terms of (i) malicious traffic identification, (ii) filtering rules cardinality, and (iii) packet processing throughput required in modern high-speed networks. The experimental results demonstrate that our signature-based approach outperforms IP-based alternatives, achieving high detection accuracy and significant generalization capabilities.

7.1 Motivation

Distributed Denial-of-Service (DDoS) attacks originate from compromised hosts and/or exploited vulnerable systems producing traffic from a large number of sources [94]. Such attacks are continuously increasing in frequency and magnitude [95].

Legacy DDoS protection mechanisms maintain statistics based on source IP or network flows to detect and ultimately mitigate malicious traffic. Maintaining flow/IP-based metrics requires data from lengthy time-windows that may hinder real-time identification of malicious traffic and the subsequent mitigation. Moreover, traditional filtering mechanisms rely on IP-based rules that increase proportionally to the number of alleged

malicious sources. In massive attacks that may include millions of source IPs [94], such a filtering approach raises scalability issues [72], [96].

To counter the shortcomings of IP-based schemes, we propose a source IP-agnostic DDoS protection mechanism that classifies and mitigates network attacks based on packet signatures, i.e. unique combinations of packet field values. Motivated by our effort on SYN Flood attacks in section 6, we consider DDoS Amplification (volumetric) attacks, commonly used to overwhelm network infrastructures. The proposed approach relies on the widely observed fact that these attacks may be characterized by a modest number of salient packet characteristics [94]. Consequently, our schema attempts to dynamically reveal related packet characteristics (signatures) and use them as filters to block the attack traffic in a scalable fashion.

In a nutshell, the proposed mechanism continuously monitors the network traffic and extracts packet signatures based on the most important features tailored to an attack vector (e.g. DNS or NTP Amplification attacks). Packet signatures are classified via supervised *Machine Learning* (ML) algorithms, appropriately trained with benign and malicious traffic, focusing on distinct packet fields (features). Malicious signatures are further subjected to a reduction process before being employed as filtering rules to expedite mitigation performance. The reduced set of signatures is finally deployed on high-performance programmable scrubbing middleboxes.

The remainder of this section is structured as follows: Section 7.2 contains background information and discusses related work; Section 7.3 offers an overview of the proposed architecture; Section 7.4 provides implementation details of the proposed Signature-based Traffic Classification and Mitigation schema; Section 7.5 provides experimental evaluations for volumetric DNS attacks regarding processing performance and detection accuracy. Finally, Section 7.6 summarizes this section and discusses future steps.

7.2 Related Work & Contributions

There are various efforts reported in the literature that attempt to classify and filter DDoS attacks. In subsections 7.2.1, 7.2.2 below, we present related flow-based and signature-based schemes accordingly. Subsection 7.2.3 emphasizes on our key contributions with regards to the state-of-the-art mechanisms.

7.2.1 Flow-based Classification and Filtering

In [61], a DDoS traffic classification schema based on a Multilayer Perceptron (MLP) was introduced. Traffic metrics related to flows and packet rates (UDP, ICMP) are collected and used as input to an MLP, tasked with classifying network traffic to benign/malicious.

In [17], an *OpenFlow* (OF) DDoS detection mechanism was presented. This collects periodically entries from OF-enabled network devices, extracts flow-related features and classifies them using Self-Organizing Maps (SOM). In [62], an SDN DDoS detection and mitigation schema was proposed. Sharp increases in the rate of Packet-In messages are considered as an indication of DDoS attacks; subsequently a mitigation pipeline is triggered. *OpenFlow* rules are collected from network devices and classified via an appropriate MLP that uses the same feature set as in [17]. Malicious flows are then blocked via appropriate mitigation entries in OF-enabled devices.

In [63], a large set of flow-related features is extracted from packets and sent to OF Controllers. These features are used as input to a Stacked Autoencoder (AE), which classifies flow as benign or malicious. Authors highlight processing limitations in Controller-based packet collection and feature extraction.

In [58], a two-level protection schema was introduced. Initially, entropy values are calculated for the number of destination IPs and ports, with sudden changes indicating an ongoing attack. The victim is identified and traffic destined towards its IP is redirected to an OF-enabled switch. This device acts as a second, more refined level of detection, that uses packet symmetry to identify malicious flows. Malicious flows are subjected to source IP-based aggregation to reduce the required blocking rules. Finally, filtering rules are deployed to the OF switch while benign traffic is redirected back.

In [64] *ATLANTIC*, an SDN framework for DDoS attack detection and mitigation, was proposed. Entropy changes for specific flow features within consecutive time-windows indicate the existence of an attack. Network flows responsible for entropy changes are fed in a traffic classification component based on K-means and Support Vector Machines (SVM). K-means is used initially to create clusters of common flows and SVM is further used to identify malicious flows. Subsequently, drop rules are installed for malicious flows.

A flow-based traffic classification mechanism was suggested in *LUCID* [66]. Flow values are collected from different time windows and represented as arrays; subsequently these arrays are fed to a Convolutional Neural Network (CNN) to identify time-dependent traffic patterns. Attack mitigation was not addressed in the *LUCID* paper.

7.2.2 Signature-based Classification and Filtering

Signature-based traffic classification and filtering is commonly featured in Intrusion Detection/Prevention Systems (IDS/IPS), e.g. *Suricata* [97]. Network packets are monitored and their packet field values are compared to predefined sets of malicious signatures. Notably, the widely employed DDoS detection tool *FastNetMon* [60], relies on static rules to identify Amplification attacks. Although these approaches are able to instantly identify previously observed attack patterns, they are not able to detect zero-day threats.

By contrast, in [98] a tool for extracting zero-day attack signatures was proposed; upon the detection of an attack, their system analyzes both benign and attack packets. Signatures suddenly appearing in high frequency in the network traffic are attack indicators, while evenly distributed signatures usually characterize benign traffic.

In [65] *DeepDefense*, a DDoS detection schema based on Recurrent Neural Networks (RNN) was introduced. Traffic traces, collected within sliding time windows, are translated into arrays of packet features. These are fed to an RNN that segregates malicious from benign packets.

Finally, Cloudflare, currently one of the largest Content Delivery Networks (CDN) that also offers DDoS protection services, employs packet signatures to filter malicious traffic [93]. To the best of our knowledge, the exact methods for traffic classification and signature-based filtering are not publicly available and thus we cannot compare our approach with them.

7.2.3 Key Contributions

Our key contributions can be summarized as follows:

- Most of the reported efforts in the literature employ metrics aggregated by IP addresses or network flows for traffic classification [17], [61]–[64]. In contrast,

we focus on the most appropriate packet features to identify malicious signatures based on *Supervised Learning* algorithms. Due to their enhanced generalization capabilities, these can accurately identify zero-day (unseen) attacks (outperforming static approaches [60]).

- We exploit common characteristics observed in the attack traffic to generate appropriate signature-based filtering rules. These are subjected to a reduction process that minimizes their number and expedites the mitigation performance.
- Our approach does not require collection of data over lengthy time-windows and corresponding time references as in [65], [98]. Instead, current packet field values are used, thus expediting detection and mitigation of attack traffic with no significant deterioration of classification accuracy.
- We propose a dynamic, tunable yet high-performance scrubbing mechanism based on programmable software data planes (XDP). Unlike proprietary monolithic solutions, our approach offers programmable monitoring and filtering functionalities without compromising on packet processing performance.
- We conducted detailed experiments focusing on volumetric DNS attacks; we employed high packet rates and real network data (benign and malicious) to illustrate the applicability of our mechanism in production network environments.

7.3 Design Principles & Architectural Overview

In this section, we outline design principles and present a baseline overview of the proposed Signature-based Traffic Classification and Mitigation architecture.

7.3.1 Design Principles

The main design principles of our mechanism are summarized below:

Signature-based filtering: We opt to surgically mitigate DDoS attacks focusing on distinct packet feature combinations (signatures) exhibited by offending traffic. Unlike traditional DDoS defense mechanisms that rely on blocking a massive number of IP sources, our approach attempts to generate IP-agnostic filtering rules.

Filtering rules reduction: Filtering rules are stored within network devices (switches, routers, firewalls) that typically impose limits to the number of entries they can support. To reduce their number, source-IP based procedures [58], [72] employ IP aggregation

techniques. Our signature reduction mechanism identifies instead a concise set of rules required to block an attack, with minimal effect on benign traffic.

Traffic classification based on supervised Machine Learning (ML) algorithms: Our approach is trained using packet characteristics from normal (benign) traces and past attack incidents. The learning process can be tailored to specific network environments, thus enhancing classification accuracy. To that end, the employed features should be carefully selected and tuned depending on anticipated attack vectors.

High performance scalable Network Functions based on programmable middleboxes: Typically, traffic monitoring and filtering functionalities are implemented by monolithic appliances. In contrast, we opted to use COTS hardware (i.e. low-cost NICs) as data plane programmable appliances powered by the XDP framework. This enables online packet handling without imposing control plane processing overhead. XDP-enabled appliances can be instantiated on-demand and scaled according to traffic and application requirements, thus providing a suitable mechanism for cloud-based scrubbing services.

7.3.2 Architectural Overview

In Figure 7.1, we present a high-level overview of the proposed architecture for DDoS protection, applicable either in transit provider networks or customer/edge network domains. Our mechanism consists of four separate components that offer: (a) *Signature Extraction*, (b) *Signature Classification*, (c) *Signature Reduction* and (d) *Anomaly Mitigation*. In what follows, we outline the DDoS detection and mitigation workflow referring to steps *i – vi* illustrated in Figure 7.1.

Benign and malicious traffic originating from various Internet sources traverses through a network infrastructure equipped with programmable devices. Network traffic is continuously monitored (step *i*) in the data plane by the *Signature Extraction* component. This component employs high-performance programmable mechanisms (e.g. XDP) to extract appropriate packet fields, i.e. signatures, pertaining to different attack vectors. Note that these fields should be selected after careful examination of benign and malicious traffic for a specific exploited protocol. Our methodology for selecting the most important packet fields (features) will be presented in subsection 7.4.1; note that the proposed method is not limited to a specific attack vector.

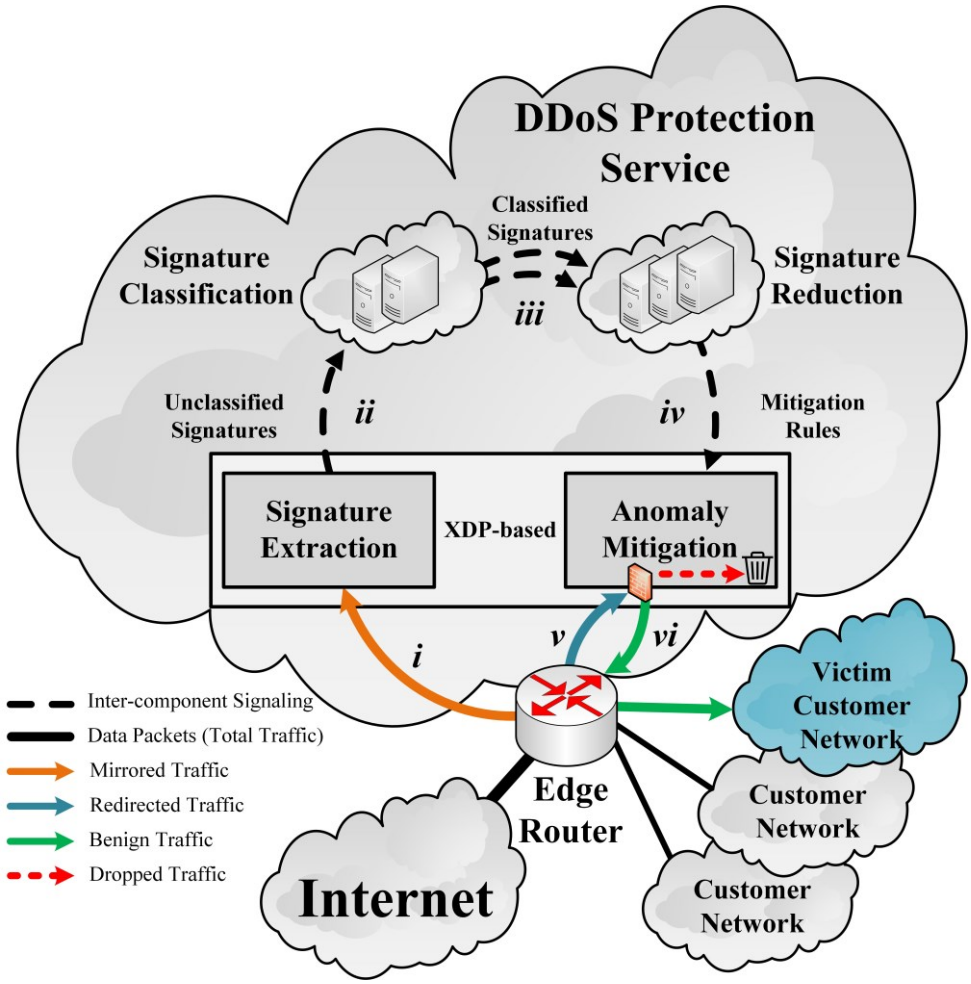

Figure 7.1: High-Level Overview of the Signature-based Traffic Classification and Filtering Architecture

Extracted monitoring data (signatures) are organized per destination IP address and relayed (step *ii*) to the *Signature Classification* component, a control plane module that categorizes them as either benign or malicious. This component relies on classification methods based on supervised ML algorithms that have been trained with attack and benign traffic. Malicious signatures identify ongoing attacks targeting specific IP addresses (victims). Classified signatures are subsequently employed for mitigation rule generation (step *iii*) via the *Signature Reduction* component that expedites mitigation performance. This reduction process is formulated as a multi-objective (Pareto) optimization problem. Specifically, combinations of the most important packet features are explored to identify a smaller feature set that minimizes the number of malicious signatures for an acceptable level of benign traffic drops. The selection of a Pareto optimal pair is based on DDoS Protection service operator preferences.

Finally, the reduced set of signatures is conveyed (step *iv*) to the *Anomaly Mitigation* component, that acts as a traffic scrubbing mechanism in the data plane. Data packets

destined to the victim IP are redirected to this component (step *v*) via appropriate traffic diversion techniques. Malicious packets are dropped while benign traffic is returned back to the router (step *vi*) to be forwarded to the destination IPs.

Extraction, classification and reduction of signatures, as well as mitigation rule generation, are performed continuously in distinct intervals (time-windows). Selected intervals should be small (e.g. 10 seconds) to enable rapid propagation of information and ultimately prompt accurate traffic scrubbing.

7.4 Packet Feature Selection & DDoS Protection Detailed Architecture

Our methodology for packet feature selection and implementation details of the components shown in Figure 7.1 are presented in the following subsections.

7.4.1 Packet Feature Selection Methodology

Packet header fields forming signatures are of paramount importance for our mechanism. They are used to (i) classify packets to malicious/benign and (ii) create filtering rules for blocking the offending traffic.

In DDoS Amplification attacks, vulnerable protocols and services are exploited in a very specific manner for generating massive amounts of traffic. This traffic exhibits packet characteristics that typically deviate from benign network traffic. In order to identify the most important characteristics pertaining to a specific attack vector, we select the relevant packet header fields (features) of each abused protocol. For that purpose, we employ the methodology described below.

We start with an initial set of n features $F = \{F_1, F_2, \dots F_n\}$, that includes (i) packet header fields of an abused protocol (e.g. DNS) and (ii) IP packet Total Length and UDP datagram Length fields (these values may differ in cases of IP fragmentation of large UDP packets). The former may reveal packet field values that are employed for generating large payloads in such attacks. The latter may correspond to large values, typical for DDoS Amplification [2].

The packet header field selection algorithm uses both benign and malicious traffic for an attack vector to train a Random Forest (RF) classifier based on a training dataset T of examples with F features. The RF training process provides (i) the Out-Of-Bag (*OOB*)

score, a metric that shows the accuracy achieved on examples that were not included in the training process of each decision tree [99] and (ii) the importance of each feature [91]. High values of *OOB* score illustrate that the employed fields can be used to accurately classify benign and malicious packets. The feature selection pseudocode is:

Packet Header Field Selection Algorithm

Input: Training Dataset T, Packet Features $F=\{F_1, F_2, \ldots F_n\}$
Output: Packet Features $F'=\{F_1, F_2, \ldots F_m\}$
1: $(OOB_n, F_{\text{ranked}}) \leftarrow$ Random Forest (T, F)
2: $F_{\text{ranked}} \leftarrow$ sort_descending(F_{ranked})
3: **for each** q ∈ [1, n) **do**:
4: m = n-q
5: F' = TOP m entries from F_{ranked}
6: $OOB_m \leftarrow$ Random Forest (T, F')
7: **if** $(OOB_n - OOB_m) \geq \varepsilon$ **then**
8: return F'
9: **end for**

The RF feature importance metric enables the selection of $m < n$ important features according to the above iterative process, see also [100]. As a result, we obtain a reduced set of features $F' = \{F_1, F_2, \ldots F_m\}$ that are used for packet monitoring, traffic classification, signature reduction and attack mitigation purposes.

The elimination of non-important features (selecting m most important ones) has the following benefits for our schema: (i) increased packet throughput of *Signature Extraction* and *Anomaly Mitigation* components of Figure 7.1 since fewer packet fields are required to be parsed and stored; (ii) enhanced accuracy and shorter training times of *Supervised Learning* algorithms; (iii) lower complexity of the *Signature Reduction* component due to the lower dimensionality of its input.

7.4.2 Signature Extraction

The Signature Extraction (SE) component is a high-performance monitoring mechanism based on the XDP framework. It collects mirrored network traffic, extracts appropriate packet fields, and conveys monitoring data to the *Signature Classification* (SC) component, as illustrated in Figure 7.2.

The combination of packet feature values can be represented by the signature vector X = $[x_1\, x_2 \ldots x_m]^T$, where x_i is the value for packet field i. Each unique signature X corresponds

to a row in the Monitoring Data Table of Figure 7.2. Every observed packet signature pertains to a counter stored within an appropriate *BPF Map* (i.e. hash table).

SE consists of various instances, each associated with a specific attack vector. They all contain a *Data Extractor* and a *Data Exporter* module:

- The *Data Extractor* is a kernel space XDP (data plane) program that extracts and stores packet header values for the preselected fields F', including the destination IP address. Destination IPs are required for the identification of the victim and subsequent traffic scrubbing (redirection and filtering).
- The *Data Exporter* is a user space program that periodically retrieves the contents (i.e. signatures) of the *BPF Map* and conveys them to the SC component.

Note that the SE component could be implemented using any approach that provides access to packet fields such as *sFlow* [74]. We opted for XDP since it provides cost-effective high-throughput monitoring of all packets (no sampling) and does not exhibit limitations on the available packet fields to be collected.

7.4.3 Signature Classification

The Signature Classification (SC) component collects monitoring data and classifies them using supervised *Machine Learning* (ML) methods to identify malicious signatures. It consists of the *Data Handler* and the *ML Classifier* module. The *Data Handler* module collects the different signatures X relayed by the SE component and preprocesses them (if needed) in a data normalization step. In turn, the set of X is used as input to the *ML Classifier* module which classifies them as benign/malicious. This module is trained with malicious and benign traffic datasets related to a specific protocol (e.g. DNS attacks and benign DNS traffic).

Malicious signatures correspond to ongoing attacks targeting specific IP addresses (victims). The mitigation process for the victim IP addresses is initiated by conveying malicious and benign signatures to the *Signature Reduction* (SR) component to generate filtering rules (see the following subsection).

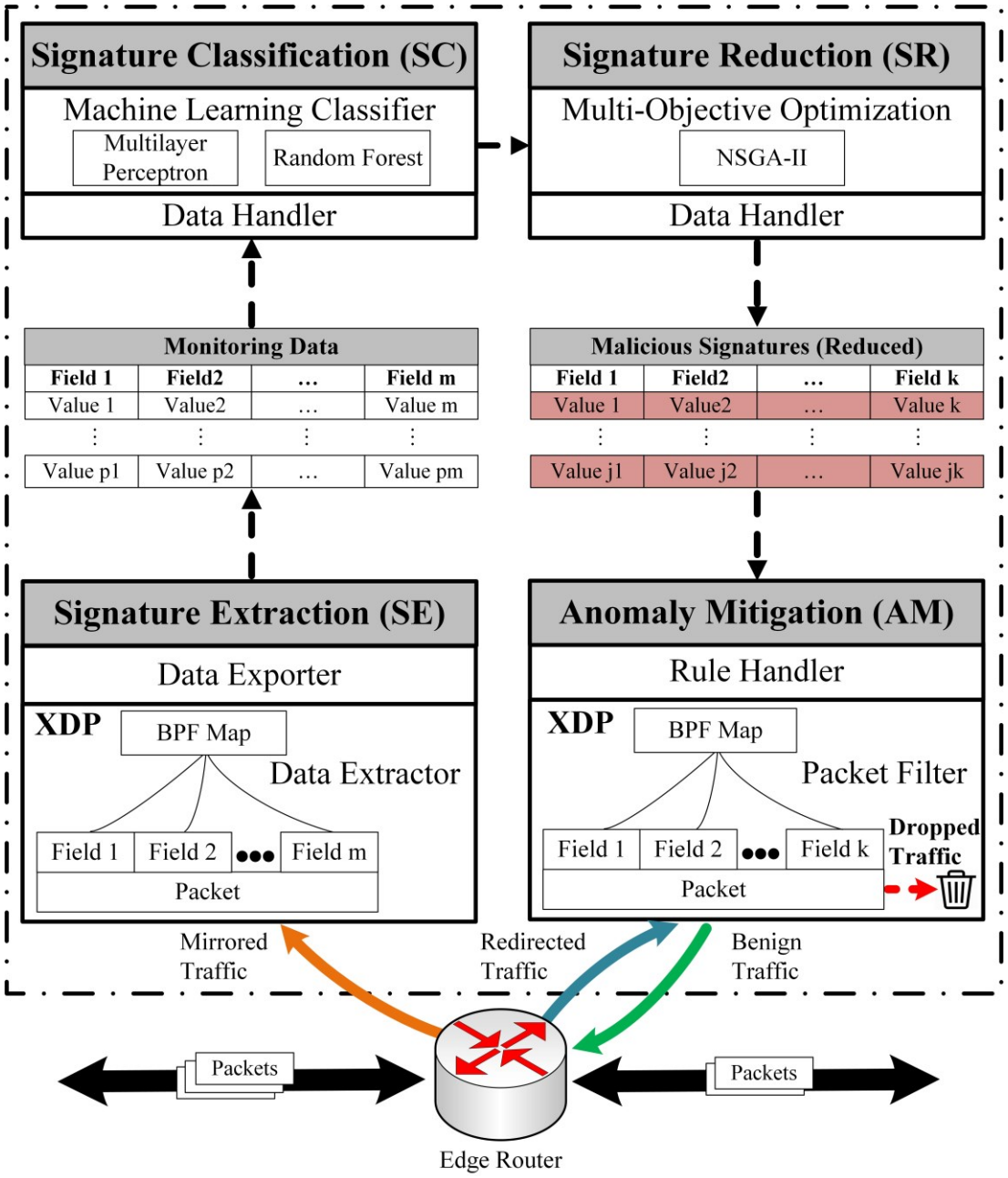

Figure 7.2: Signature-based Traffic Classification and Filtering Detailed Architecture

7.4.4 Signature Reduction

The Signature Reduction (SR) component receives both malicious and benign signatures from the SC component and reduces the number of malicious signatures to expedite the mitigation performance of the *Anomaly Mitigation* (AM) component. As mentioned, malicious signatures will be used to generate filtering rules. These are stored in memory resources (i.e. *BPF Maps* in the XDP context) that enable packet matching in the data plane. Their number significantly affects the deployment and lookup time in the *BPF Map*, which is ultimately related to the AM packet processing performance (throughput).

The SR component searches for a concise set of signatures that can block offending traffic, with minimal effect on the benign traffic. This was formulated as a multi-objective (Pareto) optimization problem, in which we search for feature subsets $F'' = \{F_1, F_2, \dots F_k\}$ of the feature set $F' = \{F_1, F_2, \dots F_m\}$, $k < m$, to identify operating points that simultaneously minimize:

(i) the number of malicious signatures (filtering rules)
(ii) the percentage of benign traffic drops

Let M' and B' be the sets of malicious and benign signatures respectively based on features from F'. For each subset F'', we similarly define M'' and B'' using only the features in F''. Objective (i) is calculated as the number (cardinality) of unique signatures in M''. Objective (ii) is the number of benign packets that correspond to the signatures in $M'' \cap B''$ divided by the number of benign packets that correspond to signatures in B''. This provides the percentage of benign traffic that would be dropped (False Positive Rate) if we used as filtering rules the signatures in M''. Note that the intersection $M' \cap B'$ is an empty set; however, the intersection $M'' \cap B''$ may result to non-empty sets in the reduced feature space F'', corresponding to False Positive cases.

The proposed optimization problem points to Pareto optimal solutions (referred to as Pareto-optimal front). However, due to stringent time constraints for attack mitigation, related algorithms would typically stop prior to Pareto-optimal front identification. We opted for a fast evolutionary approach based on Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [88]. The algorithm starts with arbitrary subsets $F'' \subset F'$ and iteratively attempts in each step to further reduce the objectives. At each iteration (generation), new subsets of F' are generated based on random combinations of F'' that correspond to the best solutions produced so far in previous iterations. The algorithm stops when a time limit is reached thus generating suboptimal subsets.

As stated above, the proposed approach will generate several solutions near the Pareto-optimal front. Naturally, only one of them can be ultimately selected for mitigating the attack. This selection should be tuned per customer network profile to depict network operator preferences, e.g. acceptable percentage of dropped benign traffic (False Positive Rate). Finally, from the selected solution, signatures of M'' are conveyed to the AM component to generate filtering rules.

7.4.5 Anomaly Mitigation

The Anomaly Mitigation (AM) component is a high-performance programmable firewall based on the XDP framework. It consists of two modules: the *Rule Handler* and the *Packet Filter*. The former receives a list of malicious signatures associated with a victim IP, installs them as filtering rules in a *BPF Map* and triggers traffic redirection for the targeted victim IP. The latter is an XDP kernel space program similar to the *Data Extractor* module of the SE component. The *Packet Filter* receives traffic destined to the victim IP and extracts the packet fields based on the reduced set of signatures F''. The extracted packet fields values are subsequently compared to the filtering rules within the *BPF Map*. If the combination of packet fields (i.e. signature) of the received packet is contained in the *BPF Map*, the packet is dropped (XDP_DROP). Otherwise, the packet is considered benign and transmitted back (XDP_TX) to the edge router to be normally forwarded to the victim IP. For implementation options related to traffic redirection and reinjection see [69].

Note that SE can be implemented with alternate monitoring solutions (e.g. *sFlow*) that can extract packet characteristics. However, the AM component is tightly coupled with programmable data planes solutions, such as XDP, able to perform inline packet filtering based on selected packet fields.

7.5 Experimental Evaluation: DNS Amplification attacks

We selected as a case study volumetric DNS attacks, one of the most common DDoS Amplification attack vectors. We evaluate our schema in an experimental testbed, employing real datasets and synthetic network traces as detailed in subsection 7.5.1 below. In short, our experiments attempt to: (i) identify and select the most important features for DNS traffic classification, (ii) assess the signature classification accuracy of *Supervised Learning* algorithms, and (iii) compare the proposed signature-based approach to source IP/flow-based alternatives.

7.5.1 Datasets Description/Testbed

Our proof-of-concept testbed is illustrated in Figure 7.3. The experimental setup was used to evaluate packet monitoring, signature classification, signature reduction, and packet filtering capabilities. The SE and AM components were implemented within the XDP

framework in the data plane. They were deployed on a physical machine (XDP-enabled node) equipped with a *Netronome* Agilio CX 2x10G SmartNIC. For packet generation purposes, we used a Virtual Machine (VM), equipped with an *Intel* X520 NIC 2x10G, able to generate packets at high rates using the PF_RING ZC framework. The SC component was implemented using the *scikit-learn* and *PyTorch* libraries while the SR component was based on the *Platypus* framework [90]. They were both deployed as control plane modules on a VM equipped with 12 vCPUs and 12GB RAM.

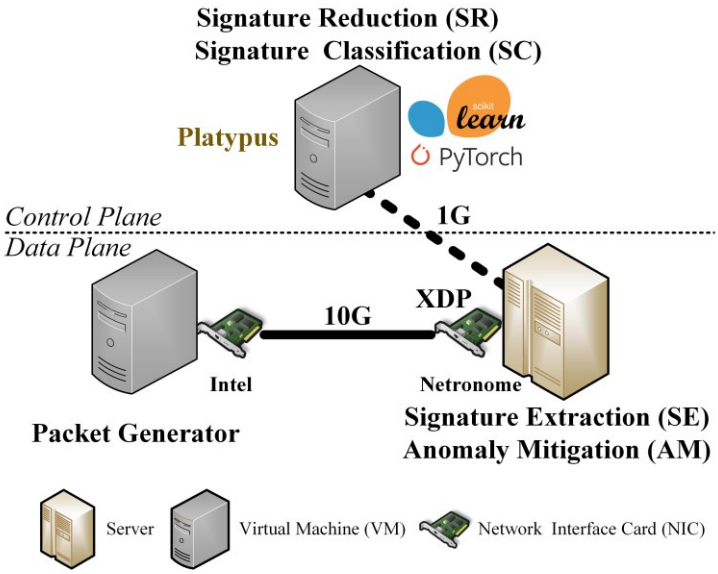

Figure 7.3: Proof-of-concep testbed setup

Real network traces were used to assess the signature classification accuracy of our schema, whereas synthesized traffic was used for stress testing packet filtering capabilities. As benign traffic, we used DNS responses from: (i) a 10G transit link between WIDE and DIX-IE (an experimental Internet Exchange), henceforth WIDE-G [80], (ii) a 1G transit link between WIDE and an upstream provider, henceforth WIDE-F [80], and (iii) Thapar University Campus Network, henceforth TU Campus [101]. As malicious traffic, we used the *Booters* datasets. These datasets, henceforth individually referred to as B_1, B_2, … B_7 or collectively as *Booters*, contain seven different DNS-based Amplification attacks generated by DDoS-for-Hire services. The attacks [2] were captured during a controlled experiment conducted between the University of Twente and SURFnet, the Dutch Research and Education Network.

All *Booters* attacks apart from B_4 and B_5 used *type ANY* DNS responses, a commonly used method for DNS Amplification attacks that returns every available Resource Record

(RR) for a given fully qualified domain name. In B_4 and B_5 attacks, the attackers attempted to use *type A* requests. Specifically, B_4 contains multiple responses for the domain *packetdevil.com*, a domain name that resolves into a very large number of IP addresses in the DNS response payload. By contrast, B_5 corresponds to a *type A* attack, that could not generate responses with heavy payload.

7.5.2 Packet Field (Feature) Selection for DNS Amplification attacks

In this subsection, we evaluate the packet header field selection algorithm for three different combinations of benign and malicious DNS traffic. Initially, we selected the 20 packet fields (features) presented in the Table below:

Table 7.1: Packet header fields (features) for DNS Traffic Classification

Packet Fields	Short Description	Packet Fields	Short Description
ip.length	IP packet size in bytes	*dns.flags. recdesired*	specifies whether recursion is desired (1) or not (0)
udp.length	UDP datagram size in bytes	*dns.flags. recavail*	specifies whether recursive query support is available (1) in the name server or not (0)
dns.id	identifies uniquely a DNS transaction	*dns.flags.z*	reserved field for future use
dns.flags. response	specifies whether the message is a query (0) or a response (1)	*dns.flags. authenticated*	indicates in a response that all data included in the answer and authority portion of the response has been authenticated by the server
dns.flags. opcode	specifies the kind of the query e.g. standard DNS query	*dns.flags. checkdisable*	indicates in a query that non-authenticated data is acceptable to the resolver sending the query (1) or not (0)
dns.flags. authoritative	specifies whether the responding DNS server is authoritative (1) or not (0) for the requested domain name	*dns.flags. rcode*	indicates the response code for the specified request e.g. the name server refused to respond

dns.flags. truncated	specifies whether the message is truncated (1) or not (0)	*dns.count. add_rr*	number of RRs in the additional records section
dns.count. queries	number of entries in the question section	*dns.qry.name*	variable length field terminated by the zero-length byte, specifying the requested domain name
dns.count. answers	number of Resource Records (RRs) in the answer section	*dns.qry.type*	specifies the type of the query
dns.count. auth_rr	number of name server RRs in the authority records section	*dns.qry.class*	specifies the class of the query e.g. IN for the Internet class

Employing the features of Table 7.1, we trained three different Random Forest (RF) classifiers consisting of 100 decision trees with default parameters of the *scikit-learn* library for tree structure and stopping [91];

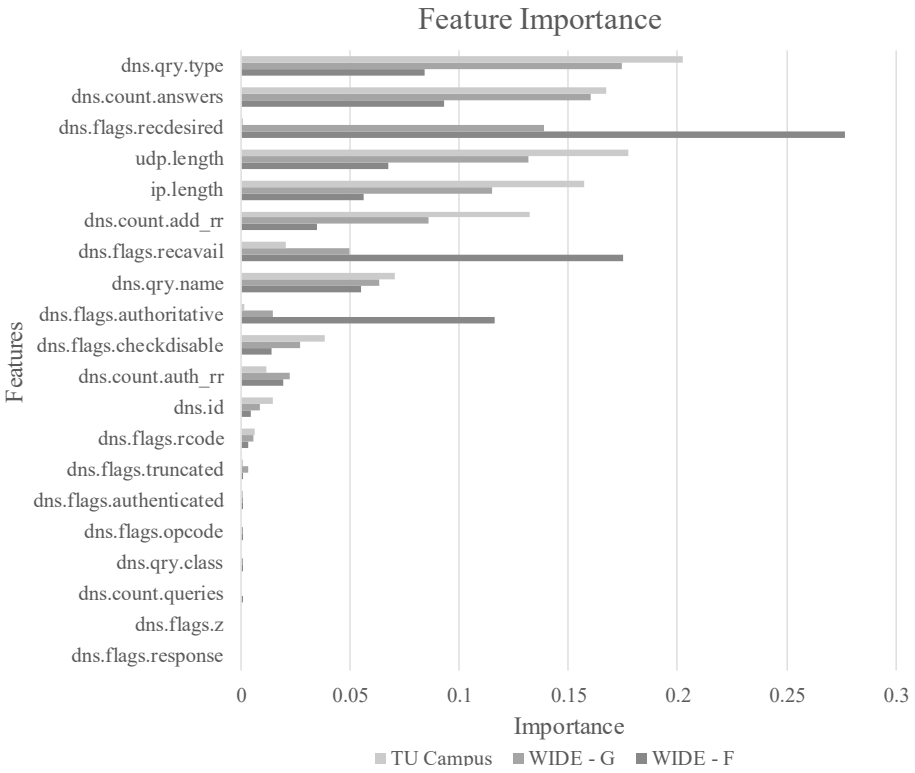

Figure 7.4: Feature Importance provided by Random Forest Classifiers for DNS Traffic

each one includes all *Booters* traffic and a particular benign dataset (WIDE-G, WIDE-F, TU Campus). The selected features except for *dns.qry.name* correspond to numerical

values and were fed directly to the RF classifiers; *dns.qry.name* was transformed to a numerical value via hash encoding. In Figure 7.4, we depict the importance of each feature for the different combinations of datasets, as computed by the *scikit-learn* library. The reported values correspond to the average feature importance for multiple training iterations.

In order to identify the most important features, we employed for each dataset combination the iterative process described in subsection 7.4.1. The threshold ε (line 7 in *Packet Header Field Selection Algorithm* pseudocode) was set equal to zero. In Table below, we present the most important features that the algorithm produced for each dataset:

Table 7.2: Most important packet fields for DNS Traffic Classification

***Booters*+WIDE-G**	***Booters*+WIDE-F**	***Booters*+TU Campus**
dns.qry.type	*dns.flags.recdesired*	*dns.qry.type*
dns.count.answers	*dns.flags.recavail*	*udp.length*
dns.flags.recdesired	*dns.flags.authoritative*	*dns.count.answers*
udp.length	*dns.count.answers*	*ip.length*
ip.length	*dns.qry.type*	*dns.count.add_rr*
dns.count.add_rr	*udp.length*	*dns.qry.name*
dns.qry.name	*ip.length*	-
-	*dns.qry.name*	-

One of the dominant features in all cases is the type of the query (*dns.qry.type*) since most attacks in the *Booters* dataset rely on DNS *type ANY* messages to generate large volumes of malicious traffic. The length of the IP packet and the UDP datagram are also important features; benign DNS traffic mainly consists of small packets while DNS Amplification attacks consist of large responses. Similarly, *dns.count.answers* and *dns.count.add_rr* can also be used to identify malicious traffic, as these counters significantly increase in attack cases. Furthermore, some of the attacks used the same *dns.qry.name* (*root-servers.net* for B_1, B_2, B_3, and *anonsc.com* for B_6, B_7) to generate large DNS packets, thus the hashed

dns.qry.name may also enhance the accuracy of the resulting classification. Interestingly, *dns.flags.recdesired*, *dns.flags.recavail*, and *dns.flags.authoritative* are of high importance for the *Booters*+WIDE-F dataset combination. This follows from the fact that most DNS responses in WIDE-F dataset (benign) were generated by iterative queries on authoritative DNS servers, while in *Booters* (malicious) by recursive queries in non-authoritative servers.

As expected, *dns.flags.response*, *dns.flags.z*, *dns.count.queries,* *dns.qry.class*, *dns.flags.opcode* are of low importance for DNS traffic classification. These had almost the same value for every packet, malicious or benign. In addition, based on our experimental observations the features *dns.flags.authenticated*, *dns.flags.truncated*, *dns.flags.rcode*, *dns.id*, *dns.count.auth_rr* and *dns.flags.checkdisable* do not improve the Out-Of-Bag (*OOB*) score of the RF classifiers and thus have been removed.

In summary, the proposed packet field (feature) selection algorithm identifies a small set of features out of the 20 initially chosen. These are used to accurately classify both benign and malicious DNS traffic patterns. The classification results are based on diverse and realistic traffic scenarios sourced from heterogeneous network environments.

7.5.3 Signature Classification Accuracy

In this subsection, we evaluate the signature classification accuracy of the proposed mechanism, using two different *Supervised Learning* methods. We implemented two classifiers: (i) Random Forests (RF) with 100 decision trees and (ii) an N x ($2N$+1) x 1 Multilayer Perceptron (MLP), with sigmoid activation functions, as suggested in [61]; N is the number of features (see Table 7.2). The MLP was trained with examples of batch size equal to 4096 and MLP weights were updated based on *Adam* method [92] with learning rate $\alpha = 0.01$. We used a single epoch with a validation dataset comprising 30% of the training dataset. The training procedure was conducted separately for each unique combination of the following:

- Each classifier (RF, MLP)
- Each benign dataset (WIDE-G, WIDE-F, TU Campus)
- Each set $A_i = \{Booters - B_i\}$, where $i = 1 \ldots 7$, e.g. $A_4 = \{B_1, B_2, B_3, B_5, B_6, B_7\}$

There are 42 different dataset combinations. Each trained model is evaluated against a mix of traffic (test dataset) based on the excluded attack dataset B_i and benign traffic from the same origin (e.g. WIDE-G). Specifically, for WIDE-G and WIDE-F, we employed two 15-minute traces for training and eight 15-minute traces as test dataset. Similarly, for TU Campus we used two 1-hour traces for training and eight 1-hour traces as test dataset respectively.

For MLP we employed undersampling techniques on the attack datasets as they contain more signatures than benign datasets. Training data for MLP were also normalized in the range of [0,1] to enhance classification capabilities. In Figure 7.5, we illustrate the *True Negative Rate* (*TNR*) of all combinations, which is the percentage of benign traffic that was classified as benign and the *True Positive Rate* (*TPR*), which is the percentage of attack traffic classified as malicious.

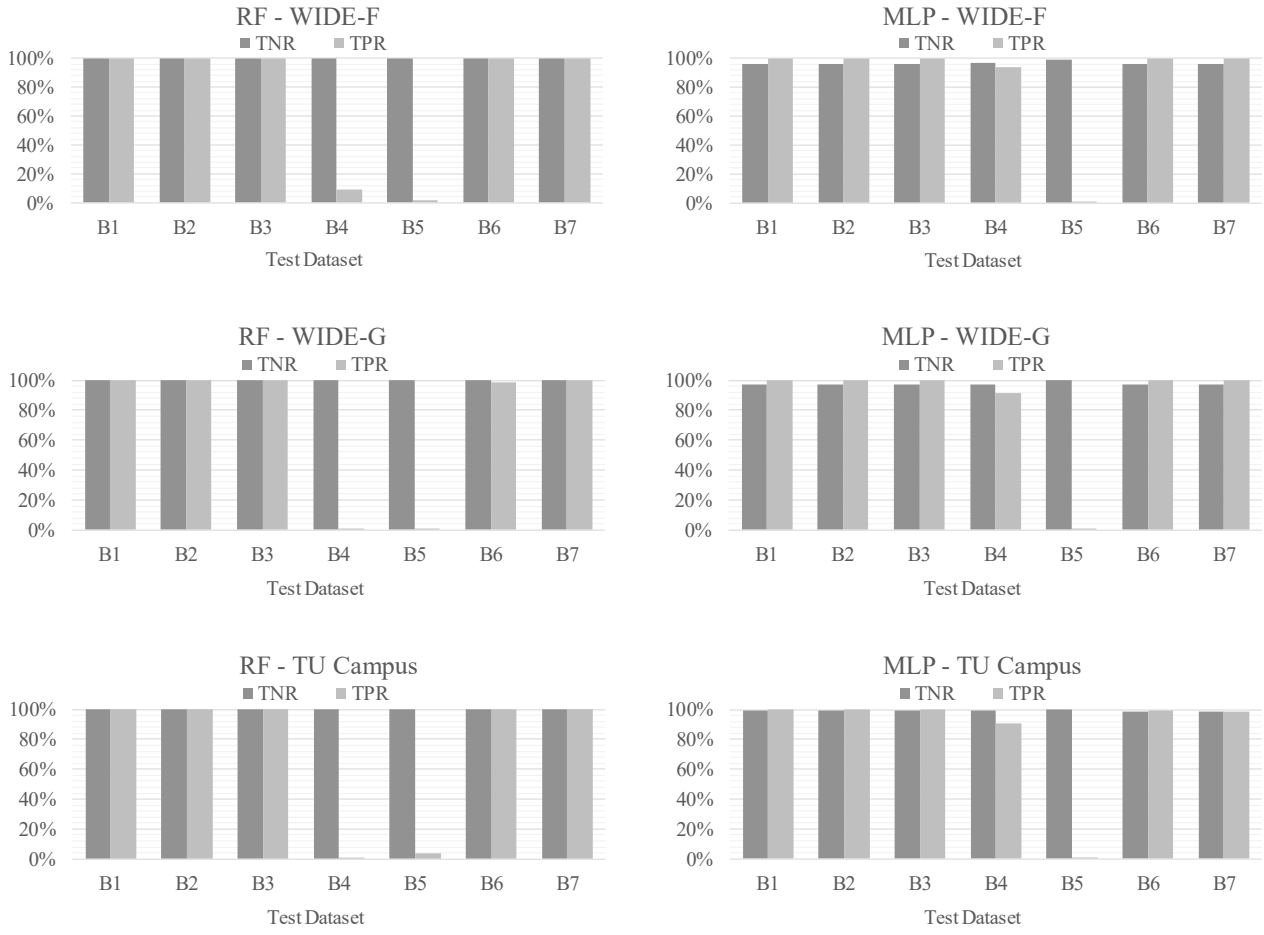

Figure 7.5: True Negative and True Positive Rates for various training scenarios using *Booters* combined with the benign datasets WIDE-F, WIDE-G and TU Campus

As illustrated in Figure 7.5, RF is a reliable method to identify both benign (WIDE-G, WIDE-F, TU Campus) and attack traffic (*Booters*) patterns, provided it is trained with diverse attack data. However, RF is not able to recognize attacks that significantly deviate from the training attack pattern. This is clearly illustrated when the model is trained with A_4, which does not include B_4 of the test dataset. Recall that B_4 contains large DNS responses with multiple *type A* RR for a domain name, while the training dataset (A_4) contains attack traces with *type ANY* DNS responses.

Similar to RF, MLP can identify benign and attack traffic with high accuracy for all combinations of training data. However, MLP identified B_4 as an attack, illustrating significant generalization capabilities on detecting "unseen" (zero-day) attacks.

Note that B_5 was not identified by any classifier as an attack trace. As already mentioned, it corresponds to a failed attack that did not produce heavy payload, thus exhibiting similarities to benign traffic. Interestingly, all classification mechanisms in our experiments discovered attack data within the benign datasets (WIDE-F, WIDE-G). A closer inspection of the original network traces revealed modest attack traffic, i.e. consecutive *type ANY* responses from specific IP sources to the same destination IP. These data were manually removed and are not included in Figure 7.5.

An interesting topic pertaining to ML algorithms are the training and test runtimes. With regards to the former, i.e. training runtime, has limited impact to our mechanism since the training process is conducted offline and the values are in any case in the order of seconds for both models. Qualitatively, training runtimes for MLP were on average 11 times faster than RF. The most important metric for us is the test runtime since it corresponds to real-time signature classification (inference). These values were in the order of milliseconds with MLP runtimes being on average 17 times faster than RFs. Such values are negligible compared to the overall time-window during which our mechanism identifies and mitigates DDoS attacks. This time-window (several seconds) includes packet monitoring, signature classification and filtering rule deployment. To our knowledge, such time-windows are consistent with production solutions offered by major security service providers.

In summary, the proposed approach provides accurate classification of DNS Amplification attacks and benign traffic. This was validated for 42 different

training/testing scenarios utilizing real data from heterogeneous network environments. Notably, MLPs achieved detection of "unseen" attack traffic patterns (not used in the training process), illustrating better generalization capabilities compared to RF classification algorithms. However, RF is still a reliable classification method, provided that it is trained with diverse attack data.

7.5.4 IP-based vs Signature-based Protection Mechanisms

In the following subsections, we compare our signature-based schema to legacy IP-based mechanisms e.g. [16], [17], [61]–[64]. We evaluate both approaches considering their (i) ability to identify and filter malicious traffic, (ii) filtering rules cardinality, and (iii) packet filtering performance.

7.5.4.1 Malicious Traffic Identification and Filtering

Typically, DDoS protection mechanisms collect monitoring data within time-windows (TW) and utilize them to classify network traffic. Based on this classification, filtering rules are generated and used to block the attack traffic. In this subsection, we compare our signature-based protection mechanism to the optimal IP-based approach, that is able to identify all malicious IP sources of an attack.

In our comparisons, we analyzed network traffic from the first time-window of each attack dataset B_i and extracted the malicious DNS signatures (based on WIDE-F features) and source IP addresses. Subsequently, we calculated from the whole attack dataset B_i the traffic (in bytes) that corresponds to the extracted DNS signatures and IP sources divided by the total attack traffic. This illustrates the percentage of the attack traffic that is blocked by each approach based on monitoring data from the first time-window of the attack. In Figure 7.6, we present for every B_i the dropped attack traffic (%) considering various time-windows and packet sampling rates. Short TWs (e.g. 1s) allow for rapid detection and mitigation. Sampling rate 1:1 corresponds to our XDP-based monitoring approach (SE), while lower values correspond to sparse packet sampling, typically employed in monitoring mechanisms e.g. *sFlow* [102].

Our signature-based approach outperforms the source IP-based alternative for all attack scenarios and combinations of time-windows (TW) and sampling rates. This is attributed to the fact that the attack traffic is characterized by a few number of DNS signatures, typically distributed to multiple IP addresses. Decreasing the sampling rate reduces

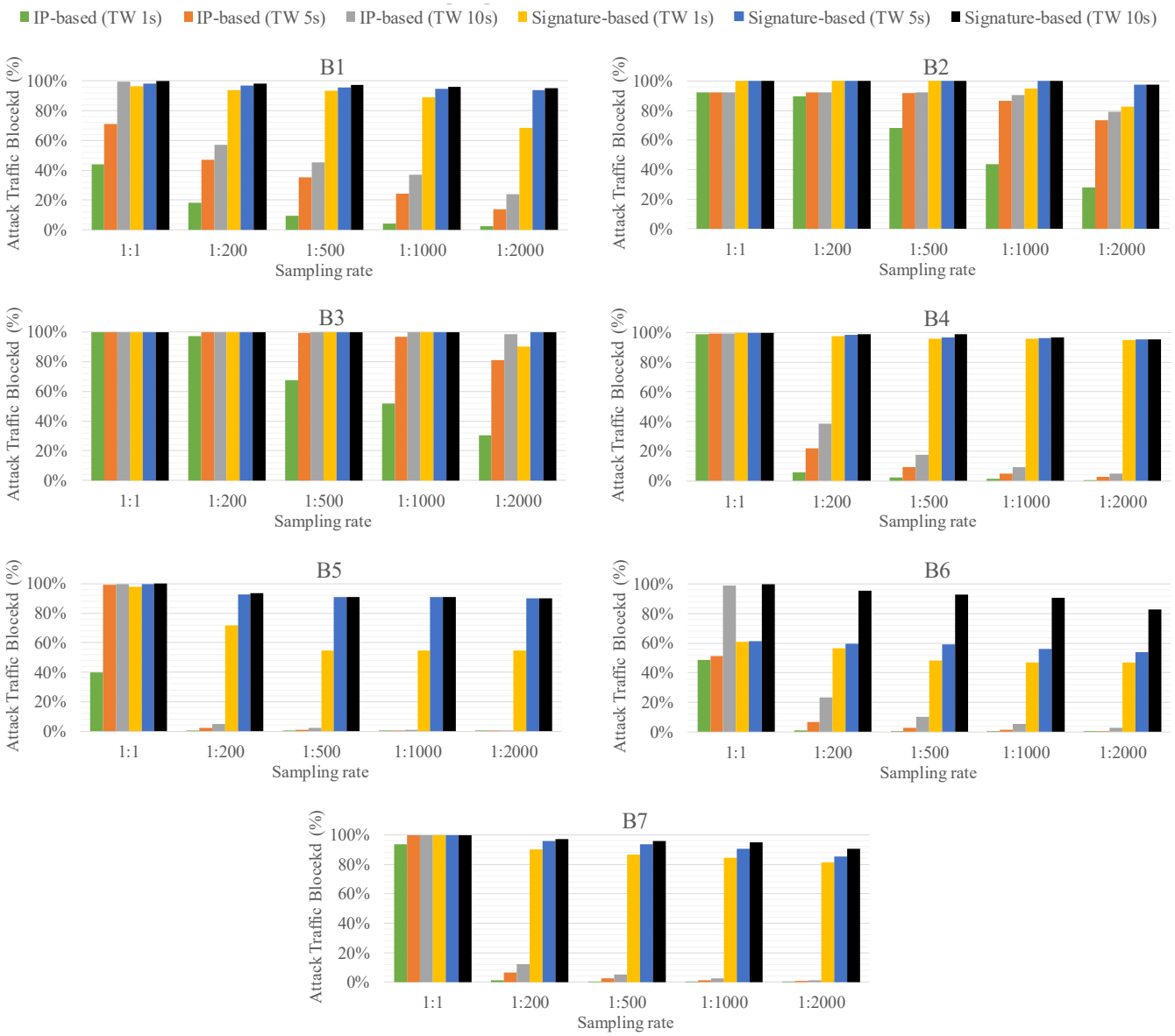

Figure 7.6: Comparison between source-based and signature-based protection mechanisms for *Booters*

significantly the effectiveness of the source-based mechanism especially for highly distributed attacks (e.g. B_1, B_4, B_6, B_7). In contrast, our approach is not affected and is able to successfully block most of the attack traffic (e.g. TW 1s - B_3: 90%) even for the lowest sampling rate 1:2000. As expected, increasing the time-window duration enables both mechanisms to observe more data and thus filter more attack traffic. Notably, our signature-based approach is able to filter a greater portion of the attack traffic (for packet sampling cases lower than 1:1) than the IP-based counterpart, while using data from shorter time-windows (grey bars – IP 10s vs yellow – signatures 1s bars). Note that, the total blocked attack traffic using WIDE-G and TU Campus feature sets is on average for all scenarios ~ 0.06% greater than WIDE-F and thus the corresponding results are not reported.

In summary, packet signatures are associated with larger amounts of attack packets compared to source IP addresses. This supports the observation that signature-based schemes may provide faster detection and more efficient filtering of DDoS Amplification attacks than conventional source IP-based mechanisms.

7.5.4.2 Filtering Rules Cardinality

In this subsection, the number of filtering rules required by IP-based schemes is compared to our signature-based approach. Specifically, we extracted the total number of unique sources for each *Booter* dataset (B_i) and the DNS signatures (WIDE-F features) that characterize all the malicious traffic. Subsequently, we employed our *Signature Reduction* (SR) component to calculate the reduced number of signatures that can match and block the malicious traffic (DNS signatures - reduced). SR, for all *Booters* and benign datasets combinations, concluded that *dns.qry.name* and *dns.qry.type* could be used to block all the offending traffic without blocking benign traffic portions.

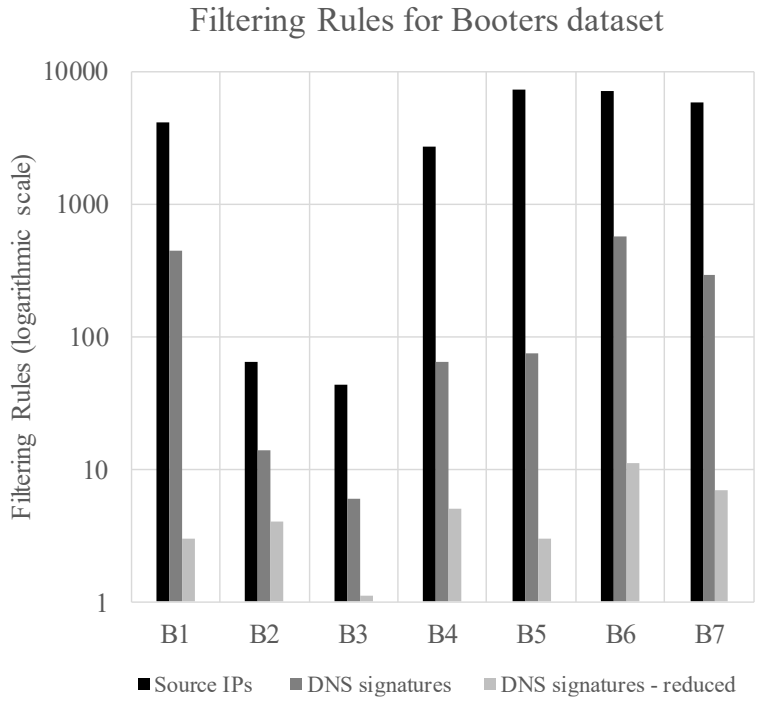

Figure 7.7: Comparison between source-based and signature-based filtering rules for *Booters*

In Figure 7.7, we compare (in logarithmic scale) the number of the source IP filtering rules to the signatures that would be required to fully block the seven DNS attacks of the *Booters* datasets without signature reduction (DNS signatures) and with signature reduction (DNS signatures – reduced).

As illustrated in Figure 7.7, the number of the required rules is decreased considerably (on average ~91% for DNS signatures and ~99% for DNS signatures – reduced). The benefits are: (i) we do not rely on source-based filters that are tough to maintain due to the extremely large cardinality of unique IPs; (ii) we are not affected by dynamic IP changes during an attack, e.g. introduced in case of rotating attackers and (iii) we significantly reduce the memory consumed in the filtering process. Note that, the total number of DNS signatures for all *Booters* using WIDE-G and TU Campus feature sets is on average ~0.6% less than WIDE-F and thus not included in Figure 7.7.

In total, our signature-based approaches require significantly less filtering rules to mitigate the total attack traffic than IP-based alternatives. As mentioned, this benefits our schema since large memory utilization results to increased lookup times in software data planes (*BPF Maps* - XDP). Hardware-based implementation may also face similar issues due to memory constraints (scarce TCAM resources).

7.5.4.3 Mitigation Performance

In this subsection, the packet filtering performance (throughput) of our approach is compared to source IP filtering alternatives. We implemented three different XDP-based mechanisms: (i) DROP_IP, an IP-based firewall that filters packets based on their source IP address, (ii) AM, that filters packets according to DNS signatures of eight features (WIDE-F features) and (iii) AM_REDUCED, that filters packets according to DNS signatures (reduced) of two features (*dns.qry.name*, *dns.qry.type*).

For stress testing, we employed synthesized network traces *DNS_TRACE*(*n)*. These contain n unique IP sources, $n/30$ unique combinations of DNS signatures of eight features and $n/850$ unique DNS signatures (reduced) of two features. The proportions were based on the experiments of the previous subsection. Note that, recent DNS Amplification attacks that targeted our University Campus, exhibited a greater proportion of IP attack sources to DNS signatures than the ones mentioned above. Thus, we anticipate that our signature-based mitigation mechanism will perform even better with network traffic profiles evolution.

We replayed various synthesized DNS traffic traces at high-speed rates (10 Million packets per second - Mpps) and measured (using the NIC drivers counters [103]) the

packets filtered by each XDP mechanism. In Figure 7.8, we present the percentage of blocked packets to the transmitted packets for various traffic traces.

DROP_IP performs better than AM and slightly better than AM_REDUCED for the *DNS_TRACE*(1,000) and (10,000); however, it faces scaling issues as the number of IP sources further increases. Specifically, DROP_IP packet processing performance (throughput) decreases from 72% to 37% as the number of IPs increases from 1,000 to 1,000,000. This validates that the number of entries in a *BPF Map* are significantly affecting its lookup time [96]. In contrast, both our AM and AM_REDUCED are scaling better in terms of packet throughput as the number of sources increases, since few DNS signatures are used to drop the attack traffic. Notably, AM_REDUCED achieves on average ~10% higher packet processing rate than the AM, presenting the added performance gain provided by reducing the number of DNS signatures. This is mainly attributed to the fewer number of entries contained in the *BPF Map* and fewer packet fields required to be parsed and processed compared to AM.

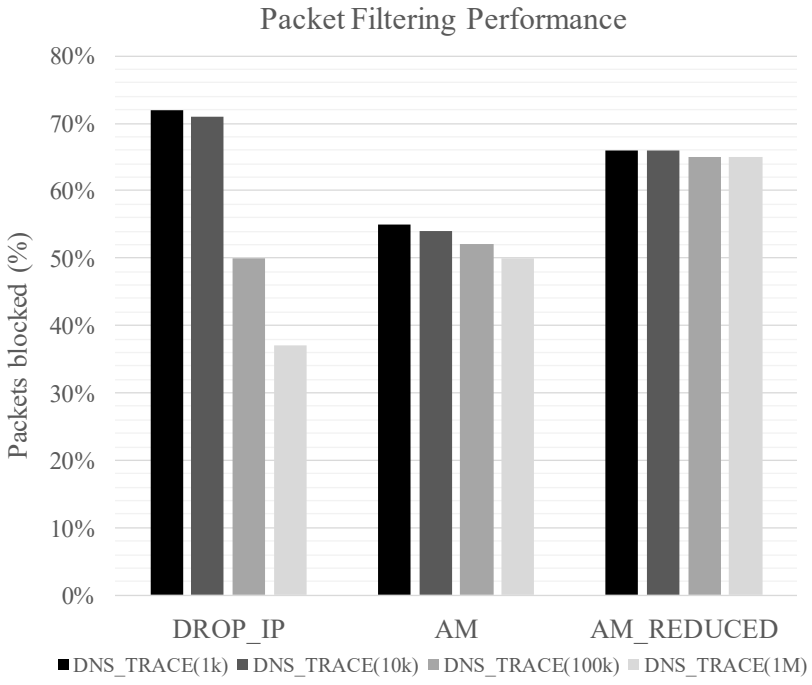

Figure 7.8: Packet throughput for IP-based and signature-based filtering mechanisms

Overall, our signature-based approach outperforms the source IP-based alternative due to the fact that the attack traffic can be described by a modest number of signatures. This is even more beneficial in massive attack scenarios where our approach achieves almost two

times greater packet filtering performance than IP-based alternatives, utilizing the same set of resources.

7.6 Summary & Concluding Remarks

In this section we presented an integrated schema for protection against volumetric DDoS attacks that employs packets signatures for traffic classification and filtering. It leverages on XDP to create performant monitoring and filtering middleboxes, tailored to different attack vectors. These operate either (i) as programmable Deep Packet Inspectors (DPI) to extract monitoring data or (ii) as flexible firewalls. Our approach does not rely on IP-sources but employs appropriate traffic signatures. This was based on the widely observed fact that volumetric DDoS attacks, especially UDP-based, may be characterized by a modest number of salient characteristics, thus enabling efficient *Machine Learning* algorithms (RF, MLP). Note that we did not consider temporal correlations since these may require network data from lengthy time-windows, thus hindering near real-time anomaly detection and mitigation.

In our proof-of-concept, we experimented with benign DNS traffic and malicious DNS Amplification attacks recorded in production network environments. The experimental results were promising and drew interesting conclusions: (i) we were able to automatically identify the most important features for DNS traffic classification for various network traffic profiles; (ii) XDP-based middleboxes were able to expediently monitor and filter network traffic; (iii) RF and MLP illustrated high classification accuracy, with the latter achieving significant generalization capabilities on detecting unknown attacks; (iv) our signature-based approach outperformed traditional IP-based schemes in terms of malicious traffic identification, filtering rules cardinality, and packet processing throughput required in modern high speed networks.

Our experimental evaluation focused on volumetric DNS attacks; however, the proposed approach is based on a generic packet feature selection methodology, and can be seamlessly extended to DDoS Amplification attacks. This follows from the fact that such attacks abuse vulnerable protocols and services in a very specific manner to generate massive amounts of traffic targeting the selected victim. Indicatively, they may exploit messages generated by SNMP *GetBulk*, NTP *monlist* and SSDP *SEARCH* requests [94]. Selecting the most important packet features (i.e. signatures) that are related to the

aforementioned attack vectors will enable implementation of protection mechanisms similar to the one proposed in this section.

Signature-based protection based on *Machine Learning* algorithms is promising for DDoS attack detection and mitigation as presented in this section. However, there are still two major challenges:

- From the perspective of a single network domain (e.g. an ISP network), the available data for training affect significantly the accuracy of the proposed classification mechanism. Thus, acquiring potentially diverse data from other (collaborating) domains (with respect to privacy restrictions) would possibly enhance the total classification accuracy.
- Despite the effectiveness of signature-based packet filtering at victims' premises, DDoS attacks may overwhelm upstream network links rendering the victim unreachable from its legitimate users. Thus, mitigating DDoS attacks in upstream networks (collaborative DDoS mitigation) before reaching the victim network would properly protect it.

Therefore, in the next section we will center on collaborative detection and cost-effective mitigation of malicious traffic across network federations.

8 Collaborative DDoS Attack Detection and Mitigation via Privacy-aware Federated Learning and Programmable Data Planes

Distributed Denial-of-Service (DDoS) attacks are delivered to their targeted victims via interconnected network domains, i.e. Autonomous Systems (AS's) of the global Internet. Although AS collaborations were instrumental in the Internet success story (e.g. global routing, peering agreements), this is largely not extended to attack protection. Collaborative DDoS detection is hindered by strict data privacy legislations while mitigation by rigid firewall solutions. In this section, we present a signature-based collaborative DDoS detection and mitigation framework. Our schema consists of a detection and mitigation application mounted in collaborating domains. The former identifies malicious packet signatures, i.e. combinations of packet field values, using Multi-layer Perceptrons (MLPs); these are cooperatively trained without exposing private data via the Federated Learning method. The latter filters malicious packets using XDP-enabled firewalls deployed in the victim AS; mitigation can also be activated on-demand within collaborating transit AS's. Our approach was evaluated both in terms of packet classification accuracy and packet processing performance using both real and synthetic network traces. The Federated Learning scheme enabled collaborators to accurately classify benign and attack packets, thereby improving individual domain accuracy without compromising privacy concerns. Collaborative on-demand mitigation is based on programmable data planes firewalls, thus providing a signature-based in-network DDoS filtering mechanism tailored to evolving federated SDN infrastructures.

8.1 Motivation

As already mentioned, Distributed Denial-of-Service (DDoS) attacks are a major threat that need to be accurately detected and rapidly mitigated. These attacks are delivered to their targeted victims via interconnected network domains, i.e. Autonomous Systems (AS's) of the global Internet.

Although AS collaborations are instrumental in the Internet success story (e.g. global routing, peering agreements), they are not extended to coordinated DDoS detection. This is mainly hindered by network operators reluctance on sharing potentially sensitive network data but also by strict data privacy legislations, i.e. GDPR [104]. *Federated*

Learning (FL) [105] is a promising approach to address such privacy concerns/regulations. It allows collaborating parties to cooperatively train *Machine Learning* (ML) models without exposing private data. FL has been proposed for various use cases like word prediction [105], healthcare applications [106] and image recognition [107]. To the best of our knowledge, few efforts [108], [109] consider collaborative DDoS detection but do not address multi-domain network environments (AS's).

In contrast to collaborative DDoS detection, collaborative mitigation has been widely employed in production network environments. Specifically, DDoS attacks are mitigated by filters enforced by collaborating AS's. These filters are typically implemented in routing devices and discard either all traffic (*BGP blackholing* [70]) or the malicious portion via a limited number of source IP/flow-based rules. In sections 6, 7, we illustrated that source IP/flow-based filtering schemes are not as effective as signature-based for DDoS mitigation. To that end, we extend the programmable firewall implemented in sections 6, 7, to provide an integrated signature-based DDoS filtering mechanism tailored to evolving federated SDN infrastructures.

Inspired by the aforementioned challenges, we extend in this section the work presented in sections 6, 7 to collaborative multi-domain network environments. Our schema detects malicious packet signatures using Multi-layer Perceptrons (MLPs); these are cooperatively trained without exposing private data. Subsequently, malicious packets are filtered in XDP-enabled [12] firewalls deployed in the victim network domain. For large-scale attacks, mitigation can also be activated on-demand in collaborating transit AS's, presumably within attack paths.

The remainder of this section is structured as follows: In Section 8.2 we discuss related efforts on collaborative DDoS protection and outline our key contributions; Section 8.3 presents a high-level overview of our mechanism and its core design principles; Section 8.4 provides implementation details for the proposed DDoS detection and mitigation framework; Section 8.5 presents experimental evaluations for DDoS detection accuracy and mitigation performance on DNS Amplification attacks. Finally, Section 8.6 summarizes our work.

8.2 Related Work & Contributions

DDoS detection and mitigation for collaborative network domains, i.e. AS's, have been widely investigated in the literature but also being employed in operational network environments. The former refers to mechanisms that allow network domains to share data for enhancing their attack detection capabilities. The latter refers to filters raised on-demand by collaborators to drop the attack traffic before reaching a victim network. Related efforts are analyzed in subsection 8.2.1 and 8.2.2 accordingly; in subsection 8.2.3 *Federated Learning* schemes for DDoS protection are presented. Finally, in 8.2.4, we present our key contributions compared to similar efforts.

8.2.1 Collaborative DDoS Detection

In [110], network traffic is monitored in disperse points of multiple network domains in an attempt to concurrently detect attacks targeting subnetworks. Attacks are identified by concurrent alerts generated by collaborating network domains. In [111], Internet Service Providers (ISPs) collaborate to detect ongoing DDoS attacks; based on predefined static rules, they exchange belief scores for suspected DDoS attacks. In [112], security events are exchanged between collaborating ISPs to validate ongoing attacks and provide appropriate countermeasures. The main focus of this work is on the communication process between collaborators. In [113], an effort for creating a European Federation of Internet Service Providers (ISPs), Internet Exchanges (IX) and Academic Networks is made; the members are exchanging attack traffic characteristics via a centralized platform without exposing victim IP addresses for privacy concerns.

8.2.2 Collaborative DDoS Mitigation

BGP blackholing [70] is the most common way for collaborative DDoS filtering. Victim networks request from upstream/peer networks to drop all traffic destined to them to protect their internal infrastructures. Although this protects network links and devices, benign traffic is also dropped. In [114], a collaborative schema for DDoS mitigation in SDN-domains is proposed. Upon the detection of the attack, specialized reports with the detected malicious sources and the victim IPs are generated; these are transferred to network domains located in the attack path, that enforce filtering rules based on the reputation of the victim domain. We extended [114] in [115], in which signaling,

coordination, and orchestration of the collaborative mitigation is based on *Blockchain* technologies; the proposed framework was tailored to federated trusted environments of wholesale network providers (Tier 1 providers) [116].

8.2.3 Federated Learning for DDoS Attacks

In [108], a DDoS detection and mitigation framework for Internet of things (IoT) environments is proposed. IoT nodes collaborate to train a common ML model via the Federated Averaging technique to accurately detect malicious traffic. This is subsequently filtered in a distributed fashion at multiple IoT nodes. In [109], a DDoS detection schema based on Federated Averaging is presented. It uses flow-based features to identify various DDoS attack types; DDoS mitigation was considered out of scope. Similarly in [117], a multi-task *Federated Learning* model is proposed. It concurrently performs DDoS detection, VPN/Tor traffic recognition and network application identification. This reduces the management overhead and the training times of individual ML models while respecting network data privacy.

8.2.4 Key Contributions

We present below how our proposed schema compares to currently suggested approaches:

- In related efforts, collaborators exchange either coarse-grained data for DDoS detection [110], [112], or predefined static rules [111], [113]; they also focus only on attack data [112]–[115]. In contrast, our *Federated Learning* scheme (i) enables for DDoS detection using both benign and attack data without exposing private information and (ii) creates ML models with generalization capabilities able to identify "unseen" (not trained with) benign and attack packets.
- Most FL schemes [108], [109], [117] simulate multi-domain data by splitting single datasets into multiple parts. Instead, we employ real network traffic aggregated by disjoint network domains, i.e. AS's, to perform fully realistic experimental evaluation.
- Typical filtering mechanisms employed in collaborative DDoS mitigation [114], [115] have the following drawbacks: they (i) support packet filtering based on predefined packet field combinations and (ii) pose limitations on the supported

number of rules. In contrast, we consider an XDP-based programmable firewall that enables packet filtering based on arbitrary packet field combinations (packet signatures) and scales its performance with the number of cores.

8.3 Design Principles & High Level Overview

8.3.1 Design Principles

We present below the core design principles of the proposed architecture:

Collaborative DDoS Detection via Federated Learning: Network traffic is classified to malicious or benign, based on *Supervised Learning* models trained via the Federated Averaging technique [105]. Thus, collaborating domains converge to *Machine Learning* configurations without sharing private data. This enables them to learn from diverse benign and attack packets without having direct access to them.

DDoS Mitigation via cloud-native scalable programmable firewalls based on the eXpress Data Path framework: We employ softwarized programmable data planes (XDP) to design high-performance Commercial off-the-shelf (COTS) firewalls for SDN infrastructures. In contrast to legacy router-based filters, these can be programmed to match and block arbitrary packet field combinations (e.g. DNS payloads as shown in Table 8.1) while scaling their resources on-demand in a vertical cloud-native fashion.

Upstream propagation of DDoS filtering requests: Our scheme enables the dissemination of filtering rules (packet signatures tailored to the attack traffic) among collaborating Autonomous Systems (cAS's). These can be used to effectively block attacks before reaching the victim AS, extending the limited filtering capabilities of *blackholing* or flow-based protection mechanisms.

8.3.2 High-level Overview

A high-level design of the proposed architecture for collaborative DDoS detection and mitigation is depicted below in Figure 8.1.

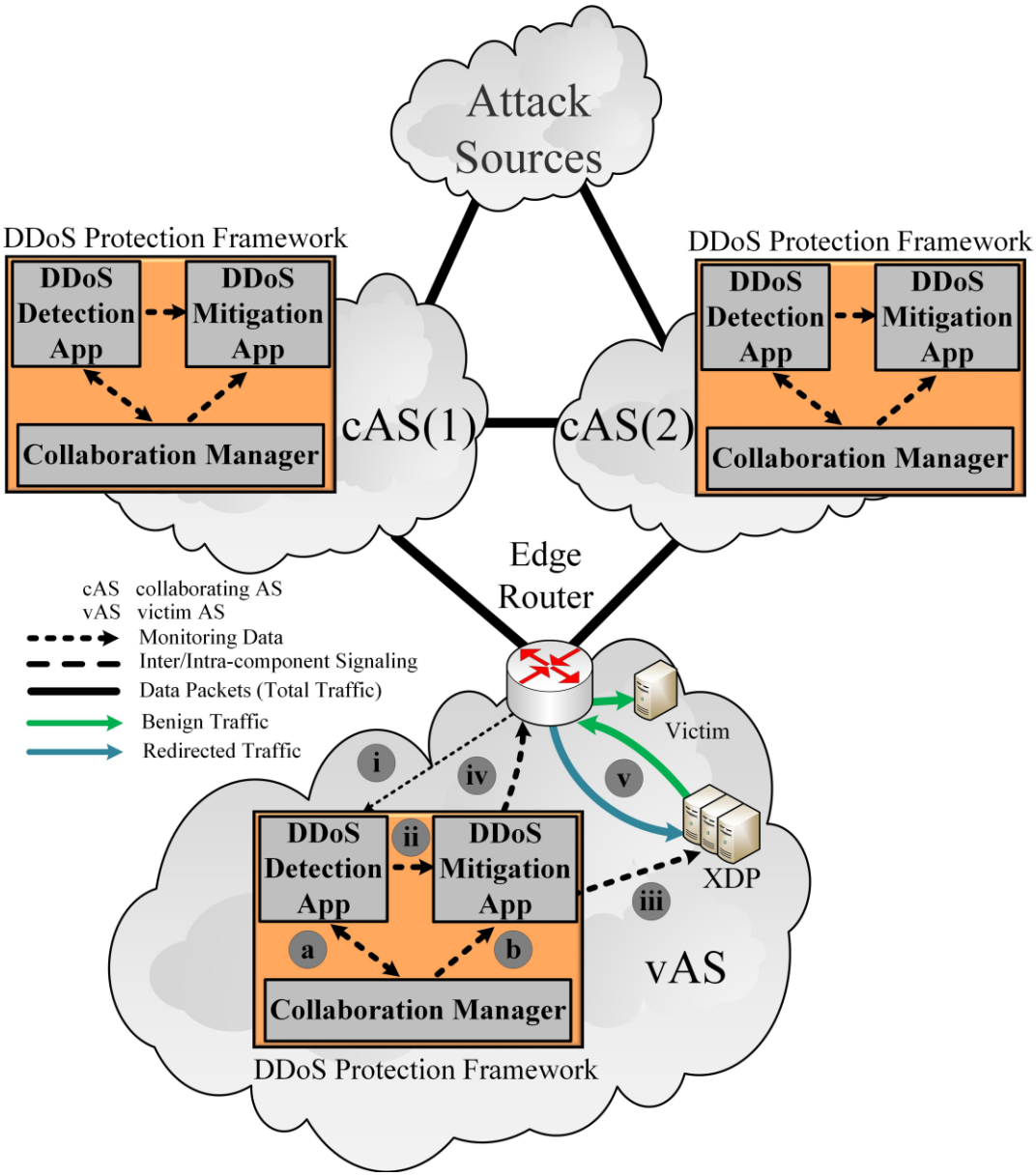

Figure 8.1: Collaborative DDoS Detection & Mitigation Architecture

Malicious actors launch DDoS attacks attempting to overwhelm the network bandwidth and/or processing resources of a host IP/subnet located in the victim AS (vAS). Both malicious and benign traffic reach vAS via interconnected Autonomous Systems, e.g. cAS(1), cAS(2). Monitoring (packet-based) data are exported by network devices (e.g. edge routers) and organized in packet signatures; these are in turn used as input to the DDoS Detection app. There, pre-trained Multilayer-Perceptrons (MLPs) classify packet signatures to malicious or benign (step i). MLPs training process has been conducted via *Federated Learning* (FL) techniques that enable distributed and privacy-preserving learning amongst collaborating Autonomous Systems (cAS's). The training process is coordinated by the Collaboration Manager (step a) in pre-agreed time-periods.

The DDoS Detection app conveys to the DDoS Mitigation app the identified malicious signatures and the corresponding victim IP/subnet (step ii). In turn, a Firewall Instance (FI) is created (step iii) that uses the identified malicious signatures as filtering rules. After FI instantiation, the DDoS mitigation app notifies the edge router to redirect traffic destined to the victim to the corresponding FI (step iv). Malicious traffic is dropped while benign traffic is bounced back and forwarded to its original destination (step v).

The DDoS Detection app based on traffic/system metrics e.g. increased link utilization, can request help from upstream/peer networks to protect its network/compute resources. The Collaboration Manager identifies adjacent cAS's that forward attack traffic [114] and populates the identified malicious signatures coupled with the victim IP address. cAS's, willing to filter malicious traffic, receive the requested signatures and signal their own DDoS Mitigation app (step b) to on-demand mitigate the offending traffic.

In our approach collaborative DDoS detection is performed in a privacy-preserving fashion without exposing collaborators private data. In contrast, collaborative mitigation requires vAS to share sensitive data, i.e. the victim IP coupled with additional specific attack characteristics (malicious signatures).

8.4 Collaborative DDos Detection and Mitigation Architecture

Our framework consists of three distinct applications (apps): (i) DDoS Detection, (ii) DDoS Mitigation, and (iii) Collaboration Manager. These are detailed in subsections 8.4.1, 8.4.2, and 8.4.3 accordingly.

8.4.1 DDoS Detection via Federated Learning

The DDoS Detection app retrieves packet-based data from external monitoring mechanisms and identifies malicious packet signatures. Signature classification is conducted by Multilayer-Perceptrons (MLPs) trained via *Federated Learning* techniques.

Monitoring data are collected within time-windows and aggregated based on preselected packet fields, forming packet signatures. Packet signatures may be represented by a vector $X = [x_1\ x_2\ \dots\ x_i]$, where x_i corresponds to packet field value i. Vectors X are used as input to Multilayer-Perceptrons (MLPs), that classify them to malicious or benign. Signatures, identified as malicious, are organized per destination IP address to generate filtering rules

at the DDoS Mitigation app of the vAS (these can also be conveyed on-demand to transit cAS's – see subsection 8.4.3 below).

The accuracy of the MLP model affects significantly the identification of malicious packets and the subsequent mitigation (since filtering rules are based on the identified malicious signatures). To improve the accuracy of the MLP model without compromising privacy, we considered a collaborative learning approach based on Federated Averaging [105].

Prerequisite for training a Federated Model (FM) is the use of a common MLP model coordinated by a neutral third party. We consider that FM may reside in a neutral independent coordinator. Such understanding is common in Internet architectures e.g. Tier-1 Providers forums [116] and major Internet eXchanges (IXes) [118].

Initially, packet fields (features) relevant to an attack vector must be selected [119]. To reduce training times and the FM complexity, inconsequential features may be eliminated. This can be achieved by not considering packet fields whose values are (i) identical in attack and benign packets or (ii) protocol specific (e.g. DNS ID, TCP sequence number). These types of features (i), (ii) are not able to enhance the classification accuracy of *Machine Learning* (ML) models and can be safely ignored upon collaborators agreements. In Section V, we evaluate our approach for DNS Amplification attacks.

Participating domains agree on common MLP hyperparameters (e.g. FM architecture, learning rates). The training process starts with an initial FM weight vector (the corresponding bias vector has been excluded for simplicity). In each iteration a new set of weights w_{FM} is evaluated and distributed amongst the k collaborating AS's[9]. Each collaborator $i = 1 \ldots k$ uses w_{FM} as initial weights and subsequently updates its local weights w_i based on its private training data N_i. These are conveyed to the FM third party coordinator that calculates the new weights w_{FM} based on the following equation:

[9] In FL, the hyperparameter k may influence the accuracy of the generated model and can be smaller than the number of all collaborators. In our experiments, this number was equal to the total number of collaborating AS's, as we did not consider a large number of participants.

$$w_{FM} = \sum_{i=1}^{k} \frac{N_i}{N} w_i, where\ N = \sum_{i=1}^{k} N_i \quad (8.1)$$

A training iteration is completed after w_{FM} calculation with new weights distributed to the cAS's. Finally, each cAS adopts the FM update that achieves the highest accuracy on its local validation dataset (subset of the total dataset not used for training but for hyperparameter selection). In case collaborators share their local accuracies per round, a common FM may be universally adopted once the (weighted) average accuracy for all participants reaches an acceptable level.

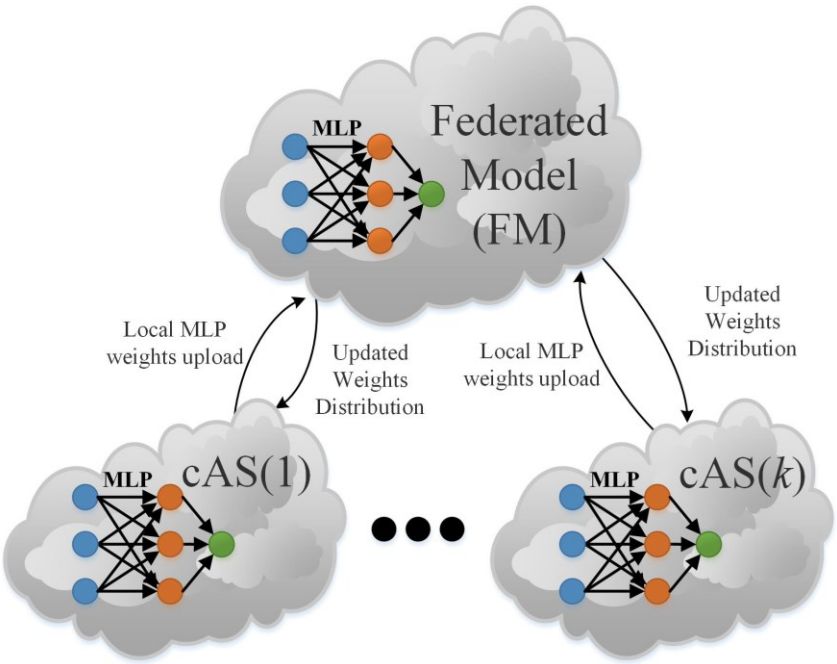

Figure 8.2: Federated Learning architecture for collaborating AS's

8.4.2 DDoS Mitigation

The DDoS Mitigation app receives requests for active attacks either from the DDoS Detection app (attacks targeting hosts of the victim AS) or the Collaboration Manager (attacks targeting other cAS's). Subsequently, this app may raise appropriate mitigation countermeasures.

Typical filtering mechanisms e.g. Access Control Lists (ACLs), *OpenFlow* (OF) rules [9], *BGP Flowspec* rules [73], are able to match and drop packets based on combinations of multiple but predefined packet fields. These rules are stored in network devices with stringent memory limitations [120]. Thus, offloading DDoS filtering to an external

firewall should (i) support any packet field combination (signature) that can match and block malicious DDoS packets, (ii) have no limit on the number of filtering rules, and (iii) allow dynamic filtering rules creation, read, update, and deletion (CRUD).

We implemented the proposed mitigation app based on the *eXpress Data Path* (XDP) framework. XDP memory structures for storing packet signatures are *Berkeley Packet Filter* (*BPF*) *Maps* (details about XDP are available in 2.2.3.1); these do not allow ternary packet field matching, i.e. the use of wildcards on packet fields. Therefore, for developing an XDP firewall program that supports various types of signatures, a *BPF MAP* per signature type would be required. This would (i) degrade the total packet processing performance due to multiple memory lookups [96], [119] (proportional to the signature types) and (ii) introduce downtime since for each *BPF Map* addition/removal, the XDP program needs to be reloaded.

The DDoS Mitigation app was designed to conform with the aforementioned XDP limitations. As depicted in Figure 8.3, it is based on a user space and a data plane program. The former manages signatures installation while the latter performs packet filtering.

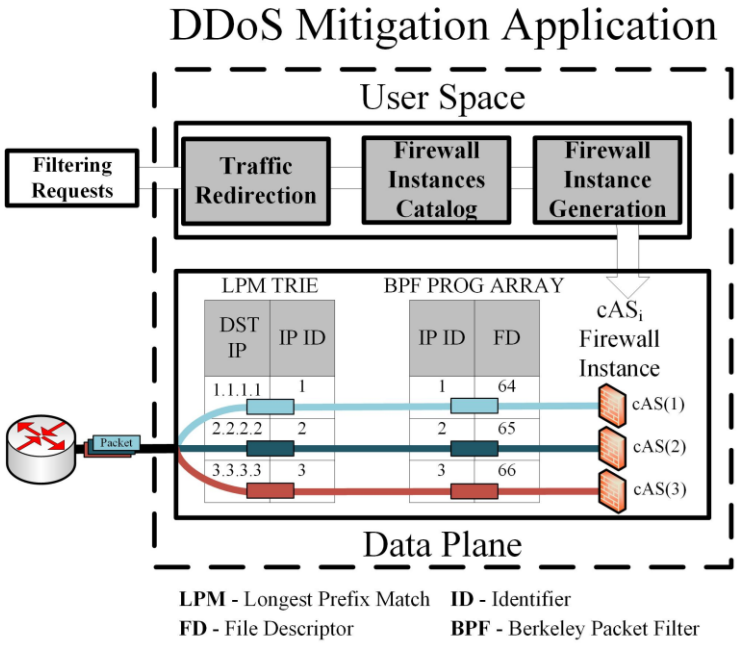

Figure 8.3: DDoS Mitigation Application Architecture

The user space program receives filtering requests from vAS and/or cAS's e.g. victim IP/network, signatures. If there are no signatures, a unique identifier *IP ID* is created (Firewall Instances Catalog). Packet signatures are transformed into XDP programs, i.e.

Firewall Instances (FIs), via appropriate *Jinja* templates [121] (Firewall Instance Generation). Each FI parses packet fields and their corresponding values that form the requested signatures. Subsequently, it contains *if-then-else* conditions to match and drop malicious packets. Each generated FI is indexed by a unique File Descriptor (*FD*) and can be accessed, updated or deleted dynamically, without affecting the packet processing operations of other FIs. After FI instantiation, the user space program signals the edge router to redirect the network traffic destined to the victim IP/subnet.

The data plane program receives the redirected packets, parses their destination IP, and performs a lookup on a *LPM* (Longest Prefix Match) *TRIE BPF Map*; this matches IP addresses/subnets to their corresponding *IP ID*. Subsequently, the *IP ID* is used as input to a special memory structure *BPF PROG ARRAY*, that passes the packet to its corresponding FI. According to the FIs signatures, malicious packets are blocked while benign packets are bounced back to the router to be appropriately forwarded.

8.4.3 Collaboration Manager

The Collaboration Manager (CM) is an application that (i) handles filtering requests for/from collaborators and (ii) coordinates the *Federated Learning* training process.

CM employs the *BGP* protocol to serialize and convey filtering requests. We needed to overcome the limitation of the predefined packet fields imposed by *BGP Flowspec*. To that end, victim's CM *BGP Speaker* initializes a *BGP* session with collaborators CM advertising the support of the *Content-URI* address family [122], similar to [114]. This allows the advertisement of specialized BGP Update messages that include *URIs* pointing to the requested filtering rules (signatures) organized in JSON representations. A filtering rule example may be found below:

Table 8.1: Signature-based filtering rule (Example)

Filtering Rule for DNS Amplification attack

```
{
 "ip_dst": "1.2.3.4/32"
 "protocol": 17
 "port": 53
 "application_protocol": "DNS"
 "payload_fields":
        {
```

```
        "dns.qry.type" = 255
        "dns.qry.name" = 0x0
        }
}
```

Note that, the use of *BGP* enables our scheme to leverage on well-established tools such as Resource Public Key Infrastructure (RPKI) to check collaborators (peers) eligibility on announcing IP prefixes/addresses.

As mentioned, CM coordinates also the Federated Averaging training process. This is an offline procedure between the collaborators and a neutral third party hosting the Federated Model. CM retrieves the generated weights from each training round and publishes them to the FM via a message broker (e.g. *RabbitMQ* [123]) . Subsequently, it receives the generated weights calculated as the average of collaborators weights. The proposed message broker scheme enables for collaborators authentication, inter-collaborators private agreements (e.g. sharing accuracy results on their local datasets) and reliable delivery of MLP weights.

Note that typical *Federated Learning* use cases [105], [108] consider as collaborating nodes low throughput devices. By contrast, in our case the total size of MLPs weights that are exchanged between cAS's have negligible impact on the high-throughput links that interconnect them.

8.5 Experimental Evaluation

We implemented all software applications of the proposed architecture and deployed them in our laboratory testbed. The DDoS Detection app was based on *pytorch* and *pysyft* python libraries. The Collaboration Manager was based on Ryu's SDN Controller *BGP Speaker* [124] and *RabbitMQ* message broker [123]. The DDoS Mitigation app was deployed on a physical machine equipped with an *Intel* i7-2600 CPU and a 10G SmartNIC *Netronome Agilio CX* [27] (XDP-enabled). This was directly connected to a Virtual Machine that offers high-speed packet generation using the *PF_RING ZC* framework [35] in a similar fashion to the testbeds employed in previous sections.

To assess the detection accuracy and mitigation performance of our mechanism, we considered DNS Amplification attacks. In subsection 8.5.1below we provide details for the employed DNS datasets. In subsection 8.5.2 we compare the classification accuracy

of the proposed Federated Model to individual (non-collaborative) approaches. Finally, in subsection 8.5.3 we showcase the packet processing performance of our mitigation mechanism.

8.5.1 Datasets Description

We focused our experiments on a commonly encountered attack vector, DNS Amplification attacks. As benign traffic, we used DNS traffic traces from a 10G transit link between the WIDE Japanese backbone and DIX-IE Internet Exchange [80]. Benign DNS traffic was aggregated per destination AS using publicly available *BGP* data [125]. In turn, AS's were sorted in descending order based on the total received packets; dataset B(i) contains benign traffic destined to AS's ranked by incoming traffic, i.e. B(1) corresponds to the AS with the highest number of DNS packets.

As malicious traffic, we used seven publicly available DNS Amplification attacks contained in the *Booters* dataset [2], henceforth referred to as A(i). Attacks in A(1), A(2), A(3), A(6) and A(7) generated *type ANY* DNS responses. By contrast, in A(4) and A(5), attackers generated *type A* DNS responses. Specifically, A(4) contains responses for a single domain name that resolved into a very large number of IP addresses. A(5) corresponds also to a *type A* attack, in which attackers could not generate responses with heavy payload. Consequently, A(5) did not succeed to generate more than few Mbps while all other attacks generated hundreds of Mbps of malicious traffic.

8.5.2 DDoS Detection Accuracy

In this subsection, we evaluate the classification accuracy of our *Federated Learning* approach and compare it to individual (non-collaborative) approaches. Specifically, we considered seven collaborating AS's, henceforth referred to as cAS(i), where

i=1…7. Each cAS(i) has access to its own private traffic mix M(i) that contains attack dataset A(i) combined with a benign dataset B(i).

We trained each cAS(i) model individually based on dataset M(i) using a Multilayer Perceptron (MLP) of 13 input neurons, 27 (13x2+1) hidden and a single output node for classification, as suggested in [61]. MLP weights were updated based on the *Adam* [92]

algorithm. The features employed for the MLP model are based on a subset of the packet fields of Table 7.1 according to the methodology presented in subsection 8.4.1:

Table 8.2: Packet fields (features) for DNS packet classification

Packet Fields (Features)	
ip.length	*dns.flags.checkdisable*
udp.length	*dns.count.answers*
dns.flags.authoritative	*dns.count.auth_rr*
dns.flags.truncated	*dns.count.add_rr*
dns.flags.recdesired	*dns.qry.name*
dns.flags.recavail	*dns.qry.type*
dns.flags.authenticated	

The Federated Model (FM) was trained using the same MLP architecture with weights conveyed from all collaborators, as prescribed by the Federated Averaging technique [105]. The hyperparameters for cAS(i) models and FM were tuned based on grid search [126], using validation datasets comprising of 30% of datasets M(i).

We evaluated the trained models using as test datasets A(i) and B(i). Each bar in Figure 8.4 illustrates the True Positive Rate – TPR, i.e. the percentage of the attack traffic A(i) that was classified as malicious by each model. Respectively, each bar in Figure 8.5 illustrates the True Negative Rate – TNR, i.e. the percentage of the benign traffic B(i) that was classified as benign. Figure 8.4 and Figure 8.5 present the accuracy individually achieved by each cAS(i) model based on its private training data M(i) and on "unseen" datasets A(j) and B(j) of other domains j ($j \neq i$). We also include the corresponding accuracy of the common Federated Model (FM). In Figure 8.6, we depict the TPR and TNR achieved by each AS(i) averaged for all datasets, A(1), A(2),…, A(7) and B(1), B(2),…, B(7) accordingly. Note that we excluded A(5) from the average TPR calculation, since it introduced insignificant malicious traffic (~ 6 Mbps).

As shown in Figure 8.5, cAS(1), cAS(2) and cAS(4) achieve high TNR for all benign datasets; however, they are not able to detect different (not trained with) attack traffic patterns, i.e. cAS(1) is not able to detect A(4), while cAS(4) is not able to detect any other attack that deviates from A(4). By contrast as depicted in Figure 8.4, cAS(3), cAS(5),

cAS(6) and cAS(7) achieve high TPR for all attack datasets, but fail to detect diverse benign DNS traffic.

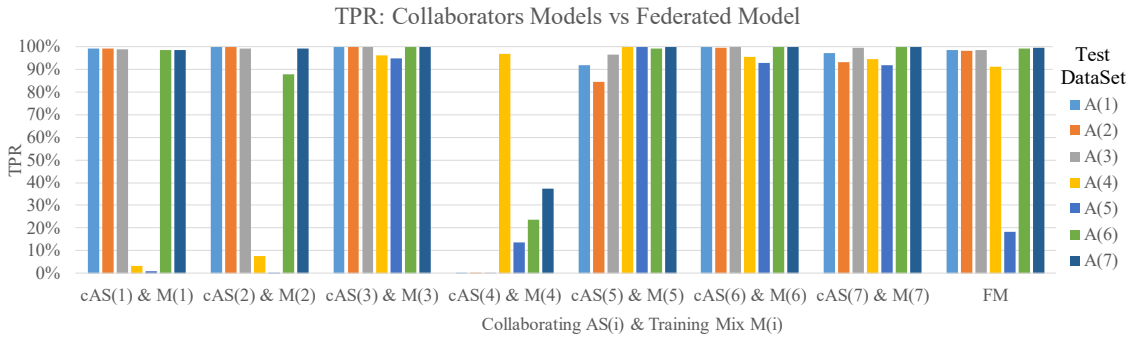

Figure 8.4: True Positive Rate for DNS Amplification attacks (Booters)

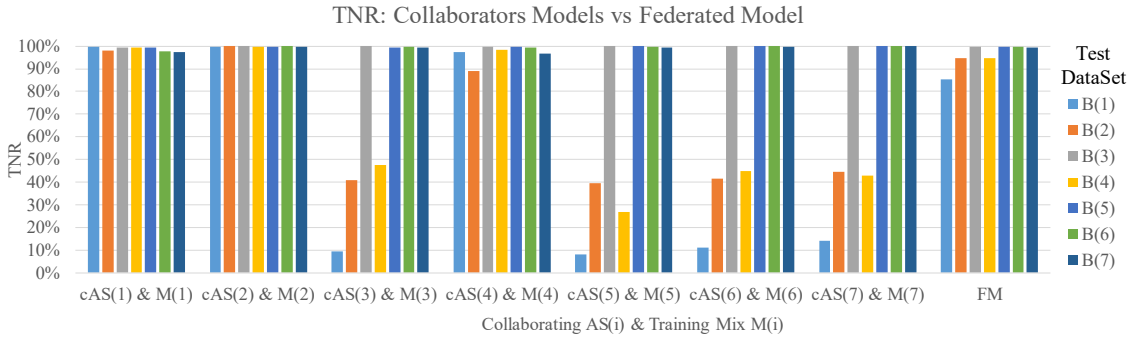

Figure 8.5: True Negative Rate for benign DNS packets

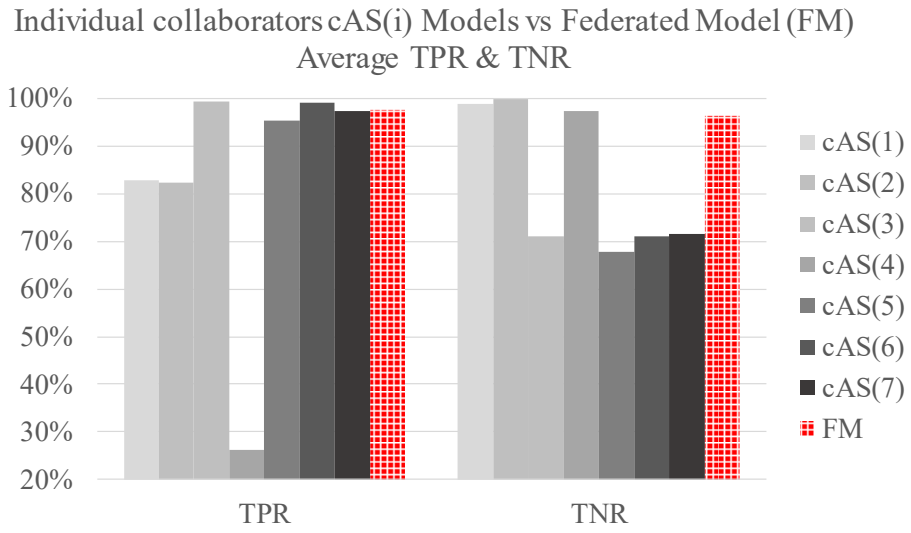

Figure 8.6: Average TPR and TNR of Individuals Models and Federated Model

The FM achieves on average the highest combination of TPR and TNR amongst individual cAS's models, as shown in Figure 8.6. Note that FM did not use private data of individual collaborators, relying only on their MLP weights. In total, the *Federated*

Learning approach enabled collaborators to identify benign and attack packets that as individuals would misclassify them.

8.5.3 DDoS Mitigation Packet Filtering Performance

In this subsection, we assess the packet filtering performance of the DDoS Mitigation app. Specifically, we evaluate the packet processing performance of our mechanism considering its CPU scalability capabilities and the number of supported Firewall Instances (FIs) within federated environments.

We generated synthesized DNS traffic consisting of packets that can be matched and dropped by a single signature per FI. This is formed by *dns.qry.type* and *dns.qry.name* packet fields based on the following condition:

if (dns.qry.type=255 and *dns.qry.name=<Root>)*

then DROP

This signature can block all the attack traffic contained in datasets A(1), A(2) and A(3). More details about our signature reduction technique, that enables us to filter attack packets using a subset of the features presented in Table 8.2, are available in 6.4.2 and 7.4.4.

We launched multiple concurrent attacks ranging from 10 to 1000 that target different collaborators with accumulated throughput of 10 Million packets per second (Mpps). To evaluate the packet processing performance, we counted the number of packets that were processed by the XDP and subsequently filtered out. This enables us to assess our firewall mechanism as a service offered to collaborating AS's. In Figure 8.7, we assess firewall scalability in terms of the deployed FIs implemented with 1, 2 or 3 CPU cores.

The packet processing performance of our mechanism scales almost linearly with the number of cores. Such behavior is also validated in [12], [77]. As expected, increasing the number of collaborators, hence the number of deployed FIs, decreases the overall packet processing rate of our firewall. Specifically, this is reduced linearly between 10 and 200 FIs and from that point it remains the same despite the increase of FIs. The enhanced performance for the small number of FIs is attributed to level one (L1) instruction cache hits while after a specific number of FIs the L1 instruction cache misses

do not affect the overall performance. These conclusions were validated using the *perf* tool [127] that provides CPU performance statistics for user-defined time intervals.

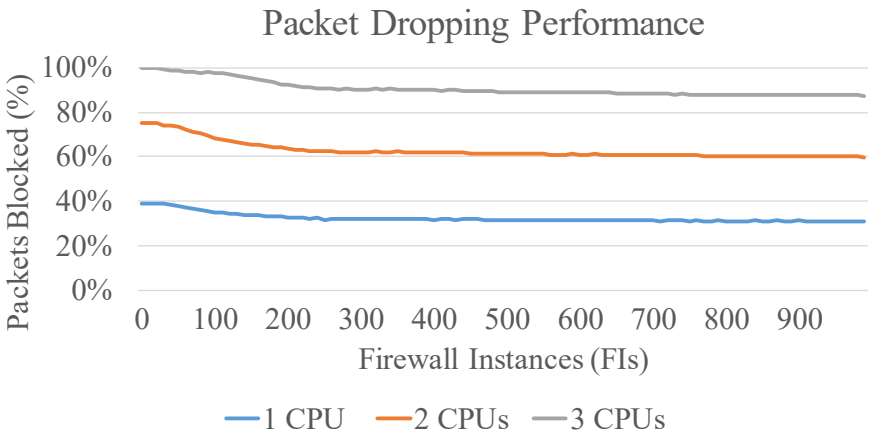

Figure 8.7: DDoS Mitigation Scalability

In total, our approach can handle successfully up to 1000 concurrent attacks targeting an equal number of collaborators. Note that these correspond to the number of concurrent blackholed IP prefixes announced in a large European IX [128]. Thus, the proposed firewall can be considered as a scalable filtering mechanism tailored to large-scale federated SDN infrastructures.

8.6 Summary & Concluding Remarks

In this section we proposed a DDoS protection framework for collaborating network domains, i.e. Autonomous Systems (AS's). Our approach leverages on the *Federated Learning* paradigm for collaborative and privacy-aware DDoS detection in SDN infrastructures. Attack mitigation is based on scalable and programmable firewalls that can be instantiated on-demand by the victim. Specifically, our schema analyzes, within time windows, packet-based data forming signatures. These are used as input to supervised *Machine Learning* models, trained cooperatively via the Federated Averaging technique. Suspicious traffic is redirected to scalable programmable (XDP-based) firewalls to be filtered out. During massive attacks, our schema enables victim AS's to raise filtering requests on collaborating AS's to block them, presumably early in attack paths.

Our framework was evaluated both in terms of detection accuracy and mitigation performance for typical DNS Amplification DDoS attacks. The conducted experiments

considered real benign and malicious network traffic. The *Federated Learning* approach enabled collaborators to accurately classify benign and attack packets improving their individual accuracy. Based on the achieved packet processing performance, the proposed programmable firewall provides a scalable filtering mechanism for evolving federated SDN infrastructures.

9 Conclusions & Future Directions

9.1 Summary & Concluding Remarks

In this dissertation, we delved into the problem of detecting and mitigating Distributed Denial-of-Service (DDoS) attacks, an everyday and of high impact problem for network operators and end-users. Specifically, we designed/implemented methodologies, algorithms, and tools for rapid detection and efficient mitigation. In a nutshell, our approach relied on emerging network technologies, i.e. high-performance programmable data planes (P4, XDP), to efficiently collect and filter DDoS attacks using intelligent data-driven algorithms to detect and classify them.

Firstly, in **Section 5**, we introduced a P4-based DDoS detection schema offloaded entirely in the data plane. In contrast to the state-of-the-art approaches that employ external network detection mechanisms (in the control plane), our approach detects network attacks within few seconds and pinpoints the under-attack subnetwork/IP. The success of our approach relies on the generation of monitoring data tailored to DDoS attacks enabled by the data plane programmability paradigm.

In **Section 6**, we made a step further towards DDoS protection, focusing on traffic classification mechanisms to segregate malicious from benign traffic. We considered SYN Flood attacks, as an indicative use case of protocol attacks, and proposed a signature-based classification and mitigation mechanism to counter them. Our approach employs packet signatures as input to *Supervised Learning* algorithms to classify network traffic. Subsequently, it generates an optimal set of filtering rules to use as countermeasure against SYN Flood attacks; these are deployed on programmable firewalls (XDP-enabled) for high-performance yet flexible packet dropping. Our mechanism illustrated high accuracy on real network traffic data and outperformed the state-of-the-art SYN Flood mitigation mechanism (*SYN Cookies*).

Inspired by the approach presented in section 6 and the challenges we faced, in **Section 7**, we extended our signature-based classification and mitigation mechanism to volumetric DDoS attacks. We relied on the widely observed fact that these attacks may be characterized by a modest number of salient packet characteristics. To that end, we employed a generic methodology for packet feature selection (signatures) and

subsequently used the most important packet fields to classify volumetric DDoS attacks. In a similar fashion to the effort described in section 6, the proposed mechanism optimizes the number of signatures required to block the attack traffic and deploys them on XDP-based programmable firewalls. Our approach was evaluated on common volumetric attacks, i.e. DNS Amplification. In our experimental evaluation, our approach identified the most important packet characteristics for traffic classification and based on them managed to accurately detect real benign and malicious DNS traffic. The proposed signature-based mechanism outperformed the state-of-the-art flow-based mechanisms in terms of traffic identification, filtering rules cardinality, and mitigation throughput.

Finally, in **Section 8** we extended the signature-based DDoS protection approach (presented in sections 6, 7) to collaborative multi-domain network environments. The proposed framework employs *Federated Learning* techniques for privacy-aware cooperative DDoS detection and incorporates a scalable yet programmable DDoS mitigation as a service mechanism tailored to collaborative network environments. Our approach was evaluated on multi-domain production network data illustrating high DDoS detection accuracy and efficient packet filtering.

9.2 Future Directions

DDoS attacks are continuously evolving to overcome the intelligent methods/algorithms/techniques employed by DDoS protection frameworks. Thus, more sophisticated methods are expected to be considered by the attackers in the future. Moreover, ever-growing attack traffic rates will be observed as the number of devices connected to the Internet is constantly increasing. To that end, DDoS protection frameworks need to consider the evolution of network attacks both in terms of scale and sophistication and be able to provide accurate and timely protection.

As illustrated in this dissertation, offloading DDoS protection tasks in the data plane is possible, featuring rapid identification and accurate mitigation of network attacks. Although appealing as a concept, packet processing limitations were faced in P4-enabled NICs. Similar hardware resource constraints are also observed in physical P4-enabled network switches [129]. Therefore, mechanisms that combine programmable (i.e. P4-enabled) hardware switches with COTS programmable (e.g. XDP-enabled) servers for DDoS detection and mitigation tasks would be of paramount interest. Especially, for

cloud-based scrubbing providers that need to offer flexible and scalable services without compromising their ability to re-program on-demand their software/hardware appliances.

With regards to network traffic classification, we showcased the existence of specific packet signatures in protocol and volumetric DDoS attacks. An interesting future direction would be the investigation of signature-based detection and mitigation approaches for application-layer attacks. The methods employed in this dissertation could be extended to incorporate the temporal characteristics of application-layer attacks. Indicatively, *Supervised* and/or *Unsupervised Learning* algorithms, e.g. Long Short-term Memory (LSTM) Neural Networks [130], Hierarchical Temporal Memory (HTM) systems [131] could be explored. Additionally, *Reinforcement Learning* techniques [132] could be also considered in cases where malicious traffic presents similarities with the benign traffic. An interesting aspect of *Machine Learning* techniques that requires further investigation for broader use in production networks is explainability/interpretability.

Finally, as mentioned in section 8, the collaboration of disjoint network domains, i.e. AS's, is crucial for protecting networks against DDoS attacks. *Federated Learning* allowed collaborating parties to exchange network metadata without revealing their actual values. However, the cooperation of multiple domains arises some interesting challenges within *Federated Learning* setups. The independent selection of diverse (potentially the most important) features by each participant (concept drift) should be incorporated to Federated models, as it depicts the knowledge of each participant for a specific task, e.g. DDoS classification. Additionally, multi-task learning architectures [117] seem promising as they enable to concurrently perform more than one tasks, e.g. to identify simultaneously more than one attack vector. This could reduce training times and the complexity of the Federated models. Finally, trust-based schemes [133] could be further investigated to improve performance, robustness, and security of *Federated Learning* schemes.

10 Extended Abstract in Greek – Εκτεταμένη Περίληψη στα Ελληνικά

Οι σύγχρονες κοινωνίες ολοένα και περισσότερο βασίζονται σε υπηρεσίες που προσφέρονται μέσω του Διαδικτύου (Internet). Ποικίλες δραστηριότητες του ανθρώπου εξαρτώνται από αυτές τις υπηρεσίες και είτε αφορούν απλές καθημερινές ανάγκες του όπως η διευκόλυνση της επικοινωνίας (π.χ. μέσα από τη χρήση των μέσων κοινωνικής δικτύωσης) είτε επεκτείνεται ακόμα και σε περιπτώσεις που αφορούν την ίδια την ανθρώπινη ζωή, π.χ. απομακρυσμένη χειρουργική. Συνεπώς, μία από τις αδιαμφισβήτητα βασικότερες απαιτήσεις που εγείρεται, είναι η εξασφάλιση της σταθερότητας και της ορθής λειτουργίας τόσο των υποδομών όσο και των υπηρεσιών που συνιστούν τον ακρογωνιαίο λίθο του Διαδικτύου.

Ένα από τα πιο συνηθισμένα και κυριότερα προβλήματα που αντιμετωπίζουν οι διαχειριστές και επηρεάζει σημαντικά τη λειτουργία των δικτύων είναι οι κατανεμημένες επιθέσεις άρνησης παροχής υπηρεσιών (Distributed Denial-of-Service attacks - DDoS). Αυτές έχουν ως κύριο στόχο τη διακοπή της ορθής λειτουργίας των διαδικτυακών υπηρεσιών (Internet services), με αποτέλεσμα να μην καθίσταται εφικτή η εξυπηρέτηση των καλόβουλων χρηστών. Οι επιθέσεις αυτές οφείλονται σε ποικίλα και διαφορετικού τύπου κίνητρα και χρησιμοποιούν ένα μεγάλο εύρος μεθόδων/τεχνικών για να πετύχουν τον σκοπό τους. Το πρόβλημα των επιθέσεων έχει πάρει σημαντικές διαστάσεις, καθώς υπάρχουν πλατφόρμες (*Booters*) που έναντι μικρού χρηματικού αντιτίμου δίνουν τη δυνατότητα εξαπόλυσης γιγαντιαίων επιθέσεων. Η κλίμακα τους είναι ικανή να θέσει εκτός λειτουργίας από μικρά επιχειρησιακά δίκτυα και κυβερνητικές υποδομές μέχρι και τεχνολογικούς κολοσσούς.

Η συνεχής ύπαρξη αλλά και η εξέλιξη αυτών των επιθέσεων έχουν οδηγήσει στην ανάπτυξη τόσο στρατηγικών όσο και μηχανισμών για την καταστολή τους. Στόχος αυτών των μηχανισμών προστασίας είναι η άμεση και με ακρίβεια ανίχνευση των επιθέσεων και εν συνεχεία η έγκαιρη αντιμετώπιση τους. Η συνεχής βελτίωση των μηχανισμών προστασίας αποτελεί βασική ανάγκη για την προσαρμογή σε νέους τύπους επιθέσεων αλλά και στην ολοένα αυξανόμενη κλίμακα τους. Βασικές απαιτήσεις των μηχανισμών αυτών είναι η ευελιξία, η απόδοση, η ακρίβεια και η κλιμακωσιμότητα. Βάσει αυτών των προδιαγραφών οι μηχανισμοί προστασίας ενσωματώνουν τόσο καινοτόμες τεχνολογίες

όσο και ευφυέστερες μεθοδολογίες για την αποτελεσματικότερη ανίχνευση και την αποδοτικότερη αντιμετώπιση των επιθέσεων.

Η άνθιση των δικτύων οριζόμενων από λογισμικό (Software-defined Networks) έθεσε νέες βάσεις στις αρχιτεκτονικές των δικτύων και κατ' επέκταση και στους μηχανισμούς προστασίας από επιθέσεις. Αρχικά, με το πρωτόκολλο *OpenFlow* (OF) δόθηκε η δυνατότητα για ενιαία κεντρικοποιημένη και ευέλικτη διαχείριση των δικτυακών συσκευών. Ειδικότερα, η δυνατότητα προγραμματισμού των δικτυακών συσκευών (στο επίπεδο ελέγχου) επέτρεπε την εγκατάσταση κανόνων προώθησης, παρακολούθησης, αλλά και αποκοπής σε συμβατικούς μεταγωγείς, δημιουργώντας πρωτότυπες αρχιτεκτονικές προστασίας από επιθέσεις.

Καινοτόμες εξελίξεις στην ανάπτυξη του υλικού (hardware) αλλά και του λογισμικού (software) των δικτυακών συσκευών, έφεραν στο προσκήνιο την τεχνολογία των προγραμματιζόμενων συσκευών στο επίπεδο δεδομένων (programmable data planes). Μέσω αυτής της τεχνολογίας δίνεται η δυνατότητα για προγραμματισμό του επιπέδου δεδομένων δικτυακών συσκευών χωρίς να επηρεάζεται σημαντικά (ή και καθόλου) η απόδοση τους. Θα μπορούσαμε να διακρίνουμε δύο βασικούς πυλώνες των programmable data planes στους οποίους δώσαμε έμφαση στην παρούσα διδακτορική διατριβή:

- τη γλώσσα P4, που προτάθηκε για τον προγραμματισμό μεταγωγέων και δικτυακών καρτών με ενιαίο τρόπο και
- το *eXpress Data Path* (XDP), μια προσέγγιση που ξεκίνησε από μεγάλους τεχνολογικούς κολοσσούς με στόχο την υλοποίηση ενός framework που επιτρέπει την ενιαία περιγραφή υψηλής απόδοσης εφαρμογών σε γενικού τύπου εξοπλισμό.

Οι παραπάνω καινοτόμες τεχνολογίες δίνουν τη δυνατότητα για συλλογή γενικού τύπου μετρικών, αλλά και ειδικών χαρακτηριστικών της δικτυακής κίνησης. Παράλληλα, προσφέρονται για σχεδιασμό και υλοποίηση υψηλών προδιαγραφών αποδοτικών μηχανισμών απόρριψης επιθέσεων.

Η έγκαιρη και με ακρίβεια ανίχνευση των επιθέσεων απαιτεί την ανάλυση της δικτυακής κίνησης σε πραγματικό χρόνο με τη χρήση κατάλληλων μεθοδολογιών/αλγορίθμων υπόδειξης ανωμαλιών. Απλές μέθοδοι στατιστικής χρησιμοποιούνταν κατά κόρον για τέτοιου τύπου αναλύσεις, ωστόσο με την ραγδαία αύξηση του όγκου των δεδομένων (big

data) αλλά και την αύξηση της πολυπλοκότητας των μοτίβων της δικτυακής κίνησης (λόγω της εξελικτικής τάσης του Διαδικτύου), οι μέθοδοι απλής στατιστικής δεν απέδιδαν τόσο ικανοποιητικά. Μεθοδολογίες εφαρμοσμένης στατιστικής και συγκεκριμένα αλγόριθμοι Μηχανικής Μάθησης έχουν πρωτοστατήσει το ενδιαφέρον για τις αποδόσεις τους σε πλείστους τομείς, π.χ. αναγνώριση εικόνων, προβλέψεις τιμών. Η ευρεία χρήση τους σε συνδυασμό με τις υψηλές ακρίβειες που πετυχαίνουν, την θέτουν σαν μια πολλά υποσχόμενη μεθοδολογία για ανίχνευση επιθέσεων και κατηγοριοποίηση της δικτυακής κίνησης.

Στόχος της παρούσας διατριβής είναι η ανάπτυξη ενός ολοκληρωμένου μηχανισμού ανίχνευσης και αντιμετώπισης επιθέσεων χρησιμοποιώντας τις δυνατότητες των σύγχρονων προγραμματιζόμενων δικτύων σε συνδυασμό με ευφυείς τεχνικές ανάλυσης δεδομένων. Στα προβλήματα που συναντώνται στην ανίχνευση και αντιμετώπιση επιθέσεων ενσωματώνονται η αποδοτική εξαγωγή δεδομένων και η ανάλυση τους, η ανίχνευση ανωμαλιών (τουτέστιν η ύπαρξη επίθεσης, η αναγνώριση του τύπου και του θύματος της επίθεσης), η κατηγοριοποίηση της δικτυακής κίνησης σε καλόβουλη και κακόβουλη και τέλος η αντιμετώπιση της επίθεσης μέσα από κατασκευή και εγκατάσταση κατάλληλων κανόνων αποκοπής.

Με γνώμονα την κατασκευή ενός ολοκληρωμένου μηχανισμού προστασίας από επιθέσεις DDoS, η συνεισφορά της παρούσας διατριβής οργανώνεται στα **κεφάλαια 5, 6, 7** και **8**, όπου συνοπτικά περιγράφονται τα κάτωθι:

- Στο **κεφάλαιο 5**, προτείνουμε ένα μηχανισμό ανίχνευσης επιθέσεων υλοποιημένο στο επίπεδο δεδομένων με τη χρήση της γλώσσας P4. Ο μηχανισμός αυτός υλοποιείται στα άκρα του δικτύου (edge devices) και αναγνωρίζει άμεσα την ύπαρξη, τους τύπους και τα θύματα επιθέσεων DDoS. Η αξιολόγηση του μηχανισμού βασίζεται σε πραγματικά δεδομένα με γνώμονα την ακρίβεια και την επίδοση του.
- Στο **κεφάλαιο 6**, προσπαθούμε βάσει των γενικών συμπερασμάτων που λαμβάνουμε από τον μηχανισμό του προηγούμενου κεφαλαίου να διακρίνουμε πιο συγκεκριμένα χαρακτηριστικά της κακόβουλης κίνησης. Χρησιμοποιώντας τις επιθέσεις πλημμύρας SYN Flood ως μία ενδεικτική επίθεση protocol-based κατασκευάζουμε έναν μηχανισμό κατηγοριοποίησης και αποκοπής της δικτυακής κίνησης βασισμένο σε ιδιαίτερα χαρακτηριστικά των πακέτων (packet

signatures). Συγκρίνουμε την προτεινόμενη λύση με τον ευρέως χρησιμοποιούμενο μηχανισμό για SYN Flood επιθέσεις, *SYN Cookies*.

- Στο **κεφάλαιο 7**, επεκτείνουμε την λογική της κατηγοριοποίησης και αντιμετώπισης επιθέσεων βάσει χαρακτηριστικών των πακέτων σε μια μεγάλη οικογένεια επιθέσεων που προκαλούν μεγάλο όγκο κίνησης (volumetric). Προτείνουμε μια μεθοδολογία επιλογής χαρακτηριστικών και εφαρμόζουμε τον μηχανισμό μας στον πιο συνηθισμένο τύπο τέτοιων επιθέσεων, τις επιθέσεις DNS Amplification. Για την αξιολόγηση της μεθοδολογίας μας, συγκρίνουμε τον προτεινόμενο τρόπο προστασίας με την κατά κόρον χρησιμοποιούμενη τεχνική που βασίζεται σε δικτυακές ροές/ διεύθυνση πηγής IP.
- Τέλος στο **κεφάλαιο 8**, επεκτείνουμε την κατηγοριοποίηση και αποκοπή επιθέσεων βάσει χαρακτηριστικών των πακέτων σε συνεργατικά περιβάλλοντα. Ο μηχανισμός που προτείνεται βασίζεται σε συνεργασίες αυτόνομων δικτυακών συστημάτων (Autonomous Systems) και κατηγοριοποιεί τη δικτυακή κίνηση, χωρίς όμως να χρησιμοποιεί τα προσωπικά δεδομένα των συνεργαζόμενων. Παράλληλα, δίνει τη δυνατότητα για αποδοτική και κλιμακώσιμη αποκοπή επιθέσεων DDoS κατ' απαίτηση των συνεργαζόμενων.

Στη συνέχεια του συγκεκριμένου κεφαλαίου θα αναλυθεί με μεγαλύτερη λεπτομέρεια η συνεισφορά της παρούσας διατριβής περιγράφοντας εν συντομία τις μεθοδολογίες που ακολουθήθηκαν στα **κεφάλαια 5, 6, 7** και **8**.

Στο **κεφάλαιο 5** παρουσιάζεται ένας μηχανισμός ανίχνευσης επιθέσεων στο επίπεδο δεδομένων βασισμένος στη γλώσσα P4. Οι συμβατικοί (legacy) μηχανισμοί ανίχνευσης βασίζονται σε πρωτόκολλα όπως το *NetFlow*, το *sFlow* ή ακόμα και το *OpenFlow*, μέσω των οποίων εξάγονται πληροφορίες σχετικές με την διερχόμενη κίνηση σε ένα δίκτυο. Οι μηχανισμοί ανίχνευσης συλλέγουν δεδομένα από δικτυακές συσκευές, τα αναλύουν και καταλήγουν σε συμπεράσματα σχετικά με την ύπαρξη επιθέσεων DDoS. Το βασικό μειονέκτημα τους είναι: (i) οι υψηλές απαιτήσεις σε επεξεργαστική ισχύ για την ανάλυση μεγάλου όγκου δεδομένων, (ii) ο υπολογιστικός φόρτος που υπεισέρχεται στην επικοινωνία μεταξύ των δικτυακών συσκευών και των μηχανισμών συλλογής δεδομένων (ιδιαίτερα κατά τη διάρκεια μιας επίθεσης) και (iii) οι περιορισμοί στους διαθέσιμους τύπους δεδομένων που παρέχονται από τον εκάστοτε κατασκευαστή/λειτουργικό. Όλα

αυτά αθροιστικά οδηγούν σε καθυστέρηση της ανίχνευσης των επιθέσεων, γεγονός που τελικά καθυστερεί και την τελική αντιμετώπιση τους.

Αντίθετα με τις υπάρχουσες προσεγγίσεις, εμείς σχεδιάσαμε έναν μηχανισμό ανίχνευσης επιθέσεων στο επίπεδο δεδομένων. Αυτός επιτρέπει την έγκαιρη ανίχνευση επιθέσεων μέσα σε λίγα δευτερόλεπτα, τον εντοπισμό του θύματος της επίθεσης και δίνει τη δυνατότητα άμεσης ενημέρωσης για εκκίνηση διαδικασιών αντιμετώπισης. Το βασικό πλεονέκτημα που δίνεται από τη γλώσσα P4, είναι η δυνατότητα προγραμματισμού των δικτυακών συσκευών ώστε να επεξεργάζονται και να συλλέγουν συγκεκριμένες μετρικές ενδιαφέροντος, που εν προκειμένω σχετίζονται με τον εντοπισμό επιθέσεων.

Ειδικότερα, καθώς διέρχεται δικτυακή κίνηση σε συσκευές (π.χ. μεταγωγείς) που υποστηρίζουν την γλώσσα P4, αναλύονται συγκεκριμένα χαρακτηριστικά της. Επιλέχθηκαν τρία βασικά χαρακτηριστικά τα οποία υποδεικνύουν την ύπαρξη επιθέσεων και μπορούν να μας συγκεκριμενοποιήσουν το εκάστοτέ θύμα της. Αυτά τα χαρακτηριστικά αξιολογούνται μέσα σε χρονικά παράθυρα, όπου αποτελούν και το διάστημα όπου αναμένεται να εντοπιστεί η επίθεση. Οι μετρικές ενδιαφέροντος που εξετάζουμε είναι:

- Η συνολική αύξηση των δικτυακών ροών, υπολογίζοντας τον τρέχοντα αριθμό τους και συγκρίνοντας τον με τον εκθετικά κινούμενο μέσο όρο του προηγούμενου χρονικού παραθύρου επαυξημένο κατά *k* φορές της αντίστοιχης απόκλισης.
- Το πλήθος των ροών ανά υποδίκτυο/διεύθυνση IP ενδιαφέροντος συγκριτικά με το συνολικό πλήθος ροών. Η αύξηση αυτής της τιμής αυτής μας υποδεικνύει ανωμαλία όσον αφορά το πλήθος ροών ανά υποδίκτυο/διεύθυνση IP.
- Τη συμμετρία κίνησης ανά υποδίκτυο/διεύθυνση IP και το πόσο αποκλίνει από την «αναμενόμενη» της συμπεριφορά.

Αν και οι τρείς μετρικές ενδιαφέροντος ξεπεράσουν όρια που τίθενται από το διαχειριστή, τότε παράγονται κατάλληλα μηνύματα από το επίπεδο δεδομένων που υποδεικνύουν την ύπαρξη επίθεσης.

Η προτεινόμενη λύση υλοποιήθηκε και δοκιμάστηκε σε πραγματικές κάρτες δικτύου *Netronome* με διεπαφές των 10Gbit. Επίσης, χρησιμοποιήθηκαν κάρτες δικτύου *Intel* για παραγωγή υψηλού ρυθμού κίνησης. Για την αξιολόγηση του μηχανισμού

χρησιμοποιήθηκαν πραγματικά δεδομένα επιθέσεων καθώς και καλόβουλη κίνηση από ένα Internet Exchange στην Ιαπωνία. Στόχος ήταν να εξετάσουμε την ακρίβεια αλλά και την επεξεργαστική απόδοση του μηχανισμού. Ο προτεινόμενος μηχανισμός πέτυχε υψηλές ακρίβειες ανίχνευσης της τάξης του 95% για επιθέσεις διαφορετικής κλίμακας. Παράλληλα, η επεξεργαστική του δυνατότητα ήταν επαρκής για δικτυακή κίνηση υψηλών ταχυτήτων (2 εκατομμύρια πακέτα το δευτερόλεπτο). Αθροιστικά αυτές οι δύο πτυχές καθιστούν την προσέγγιση μας κατάλληλη για ανίχνευση επιθέσεων σε σύγχρονα δικτυακά περιβάλλοντα.

Συνολικά στο **κεφάλαιο 5**, κατασκευάσαμε ένα μηχανισμό ανίχνευσης επιθέσεων στο επίπεδο δεδομένων με χρήση της γλώσσας P4. Ο μηχανισμός μας χαρακτηρίζεται από έγκαιρους χρόνους ανίχνευσης δίνοντας τη δυνατότητα για άμεση αντιμετώπιση των επιθέσεων DDoS. Παράλληλα, συνοδεύεται από υψηλές ακρίβειες με μικρό αριθμό από ψευδοθετικά ποσοστά (False Positive Rates), ενώ βάσει της απόδοσης του είναι κατάλληλος για σύγχρονα δικτυακά περιβάλλοντα. Όσων αφορά την ανίχνευση επιθέσεων DDoS, ο μηχανισμός που σχεδιάσαμε αποτελεί ένα πρώτο βήμα για να εντοπίσουμε την ύπαρξη επίθεσης, τον τύπο της καθώς και το θύμα που αυτή στοχεύει. Ωστόσο για να καταφέρουμε να αντιμετωπίσουμε τις επιθέσεις καθίσταται αναγκαίο να εισχωρήσουμε σε πιο λεπτομερή ανάλυση της δικτυακής κίνησης, διακρίνοντας την καλόβουλη από την κακόβουλη και στη συνέχεια να αναπτύξουμε τα κατάλληλα φίλτρα για την αποκοπή της επίθεσης.

Για την κατηγοριοποίηση της δικτυακής κίνησης αλλά και την αποκοπή του κακόβουλου μέρους της καλούμαστε να εισχωρήσουμε σε πιο λεπτομερή ανάλυση των χαρακτηριστικών της κίνησης σε σύγκριση με τις γενικές μετρικές που χρησιμοποιήσαμε στο **κεφάλαιο 5**. Στο **κεφάλαιο 6**, επιλέξαμε να ασχοληθούμε με μια από τις κλασικότερες επιθέσεις DDoS, την SYN Flood, που αποτελεί σημαντικό πρόβλημα για τις σύγχρονες δικτυακές υποδομές. Στην επίθεση SYN Flood οι επιτιθέμενοι στέλνουν μαζικά πακέτα TCP SYN σε κόμβους θύματα κατασπαταλώντας τόσο τους πόρους των ίδιων αλλά και ενδιάμεσων δικτυακών συσκευών, π.χ. δρομολογητές (routers) ή τείχη προστασίας (firewalls). Η κύρια προσέγγιση ανίχνευσης των επιθέσεων αυτών βασίζεται στη χρήση δικτυακών ροών (network flows), ωστόσο λόγω της τεχνικής απόκρυψης της διεύθυνσης πηγής (source IP spoofing), η ακριβής σκιαγράφηση των επιτιθέμενων κρίνεται εξαιρετικά δύσκολη. Πέραν της ανίχνευσης, το βασικό πρόβλημα έγκειται στην

αντιμετώπιση της επίθεσης, καθώς η αποκοπή των επιτιθέμενων βάσει της source IP δεν είναι εφικτή λόγω IP spoofing ή λόγω γιγαντιαίων λιστών από διευθύνσεις IP. Ο κύριος μηχανισμός που χρησιμοποιείται για την αντιμετώπιση τους βασίζεται στα *SYN Cookies*, μία τεχνική που κατασκευάζει κατάλληλα διαμορφωμένα μηνύματα SYN ACK που έχουν ως σκοπό να επιβεβαιώσουν την source IP του αρχικού πακέτου SYN. Παρότι ο μηχανισμός αυτός είναι αποδοτικός και προστατεύει τα θύματα από την κακόβουλη κίνηση, απαιτεί σημαντικό πλήθος πόρων για την κατασκευή των μηνυμάτων SYN-ACK ενώ παράλληλα δημιουργεί αντίρροπη κίνηση ίση με την επίθεση. Αυτό αν αναλογιστούμε μεγάλες επιθέσεις μπορεί να δημιουργήσει περαιτέρω συμφόρηση αντί να εξομαλύνει το πρόβλημα.

Παρατηρήσαμε ότι οι επιθέσεις αυτές εμφανίζουν συγκεκριμένα μοτίβα/χαρακτηριστικά στα πακέτα, δηλαδή είδαμε τη χρήση συγκεκριμένων τιμών σε διάφορα πεδία των πακέτων τα οποία ορίζονται ως signatures. Αυτή η συμπεριφορά μπορεί να οφείλεται είτε σε χρήση στατικών τιμών σε υλοποιήσεις κακόβουλων (hackers) είτε σε προκαθορισμένες τιμές προγραμμάτων αποστολής κίνησης. Επομένως, σκεφτήκαμε να κατασκευάσουμε έναν μηχανισμό που κατηγοριοποιεί την κίνηση TCP και την αποκόπτει χρησιμοποιώντας αυτά τα ιδιαίτερα χαρακτηριστικά τους. Ο μηχανισμός χρησιμοποιεί Επιβλεπόμενη Μάθηση (*Supervised Learning*) για να κατηγοριοποιήσει την κίνηση σε καλόβουλη και κακόβουλη. Στη συνέχεια κατασκευάζει κατάλληλους κανόνες αποκοπής που περιγράφουν συνεκτικά την κακόβουλη κίνηση. Τέλος, οι κανόνες εγκαθίστανται σε υψηλής απόδοσης προγραμματιζόμενα τείχη προστασίας που βασίζονται στο framework XDP. Ένα από τα πλεονεκτήματα που μας δίνει η δυνατότητα προγραμματισμού στο επίπεδο δεδομένων είναι η αξιοποίηση του μηχανισμού SYN Cookies ως εναλλακτική λύση για περιπτώσεις κακόβουλων signatures που δεν μπορούν να εντοπιστούν.

Για να εξετάσουμε τις δυνατότητες της προτεινόμενης λύσης κατασκευάσαμε ένα testbed υψηλών ταχυτήτων που απαρτίζεται από προγραμματιζόμενες κάρτες δικτύου (XDP) των 10Gbit καθώς και από κάρτες *Intel* με δυνατότητα αποστολής πακέτων σε υψηλούς ρυθμούς. Τα δεδομένα πειραματισμού μας βασίστηκαν σε πέντε πραγματικές επιθέσεις που καταγράψαμε εντός του δικτύου παραγωγής του Ε.Μ.Π., τις οποίες τις αναμείξαμε με καλόβουλη κίνηση από το δίκτυο WIDE της Ιαπωνίας. Στόχος μας ήταν να εξετάσουμε την ακρίβεια κατηγοριοποίησης της κίνησης, τις δυνατότητες μείωσης των

signatures καθώς και την απόδοση του μηχανισμού στην αποκοπή επιθέσεων σε σύγκριση με τον μηχανισμό των *SYN Cookies*.

Ο προτεινόμενος μηχανισμός κατάφερε να εντοπίσει με μεγάλη ακρίβεια τόσο τις επιθέσεις όσο και την καλόβουλη κίνηση. Παράλληλα, κατάφερε να μειώσει σε σημαντικό βαθμό το πλήθος των κανόνων που απαιτούνται για την αποκοπή των επιθέσεων. Το μικρό αυτό πλήθος μας επέτρεψε να αυξήσουμε κατά δύο φορές την επεξεργαστική δυνατότητα του προγραμματιζόμενου μηχανισμού αντιμετώπισης σε σύγκριση με τη προσέγγιση *SYN Cookies*.

Με βάση τα συμπεράσματα αυτά αναρωτηθήκαμε αν η τεχνική που βασίζεται σε signatures μπορεί να γενικευτεί και να χρησιμοποιηθεί και σε άλλου τύπου επιθέσεις όπως οι volumetric. Επίσης, κρίθηκε αναγκαία η σκιαγράφηση ενός μεθοδικού τρόπου επιλογής μόνο των σημαντικών χαρακτηριστικών της κίνησης για την κατηγοριοποίηση της. Τέλος, ήταν επιθυμητή η σύγκριση της προτεινόμενης προσέγγισης με τους de facto μηχανισμούς ανίχνευσης και αντιμετώπισης επιθέσεων που αναγράφονται στην βιβλιογραφία (αλλά και που χρησιμοποιούνται σε πραγματικά περιβάλλοντα), οι οποίοι βασίζονται σε δικτυακές ροές (διεύθυνση πηγής).

Βάσει των προκλήσεων που αναφέρθηκαν, στο **κεφάλαιο 7** επεκτείναμε τη δουλειά μας με τα signatures και σε ένα άλλο μεγάλο σύνολο επιθέσεων, τις volumetric επιθέσεις. Οι επιθέσεις αυτές βασίζονται στην ακόλουθη τεχνική: κακόβουλοι (hackers) στέλνουν κατάλληλα κατασκευασμένα μηνύματα σε κόμβους (reflectors) που φιλοξενούν συγκεκριμένους τύπους υπηρεσιών, π.χ. LDAP, DNS, MEMCACHED, με αποτέλεσμα αυτοί με τη σειρά τους να βομβαρδίζουν το επιλεχθέν θύμα με μεγάλο πλήθος και όγκο πακέτων. Οι κλασικοί μηχανισμοί προστασίας απέναντι σε αυτές τις επιθέσεις βασίζονται στην κατηγοριοποίηση δικτυακών ροών σε κακόβουλες ή καλόβουλες και στη χρήση της αντίστοιχης διεύθυνσης πηγής IP ως αναγνωριστικό για την αποκοπή της επίθεσης. Η συλλογή, αποθήκευση και ανάλυση των ροών καθυστερεί αρκετά την κατηγοριοποίηση της κίνησης, που τελικά καθυστερεί και την αντιμετώπιση της επίθεσης. Παράλληλα, η χρήση γιγαντιαίων λιστών από κακόβουλες IP για την αποκοπή της επίθεσης εμφανίζει προβλήματα κλιμακωσιμότητας τόσο στην χρήση τους σε πραγματικό εξοπλισμό όσο και στη διαχείριση τους.

Για αυτό στο **κεφάλαιο 7** σχεδιάσαμε έναν μηχανισμό που βασίζεται στα ιδιαίτερα χαρακτηριστικά που εμφανίζουν τα κακόβουλα πακέτα (δηλαδή τα signatures), τα οποία χρησιμοποιούνται για την κατηγοριοποίηση και αποκοπή της κακόβουλης κίνησης. Ο μηχανισμός μας συλλέγει μόνο τα σημαντικά χαρακτηριστικά για κατηγοριοποίηση των signatures χρησιμοποιώντας και πάλι τεχνικές Επιβλεπόμενης Μάθησης (*Supervised Learning*). Με αυτόν τον τρόπο προσφέρει άμεση ανίχνευση της κακόβουλης κίνησης κατευθείαν από τις επικεφαλίδες των πακέτων. Βάσει της κατηγοριοποίησης, κατασκευάζονται φίλτρα αποκοπής, που δεν βασίζονται στην διεύθυνση πηγής (source IP-agnostic) αλλά στα πεδία που ομαδοποιούν με συνεκτικό τρόπο την κακόβουλη κίνηση. Αυτό έχει ως αποτέλεσμα τη σημαντική μείωση του πλήθους των κανόνων απόρριψης που αυξάνει συνολικά την επεξεργαστική απόδοση του μηχανισμού αποκοπής. Η συλλογή των κατάλληλων πεδίων των πακέτων καθώς και η αποκοπή της κακόβουλης κίνησης υλοποιήθηκαν χρησιμοποιώντας γενικού τύπου (Commercial-Off-the-Shelf) εξοπλισμό, κατάλληλο για σύγχρονα περιβάλλοντα υπολογιστικού νέφους (cloud computing).

Για να αξιολογήσουμε τον προτεινόμενο μηχανισμό επιλέξαμε τις επιθέσεις DNS Amplification, αφού αποτελούν έναν από τους πιο ευρέως χρησιμοποιούμενους τύπους volumetric επιθέσεων. Στόχος του πειραματισμού μας ήταν να δείξουμε τη δυνατότητα του προτεινόμενου μηχανισμού να επιλέγει χαρακτηριστικά των πακέτων σημαντικά για την κατηγοριοποίηση των signatures, να διακρίνει με ακρίβειά καλόβουλα από κακόβουλα signatures και τέλος την απόδοση του μηχανισμού μας έναντι κλασικών μηχανισμών προστασίας που βασίζονται σε δικτυακές ροές. Για την αξιολόγηση όλων των παραπάνω, χρησιμοποιήσαμε 7 επιθέσεις DNS, καταγεγραμμένες από το πανεπιστήμιο του Twente σε συνεργασία με το ολλανδικό NREN SurfNET, ενώ ως καλόβουλη κίνηση χρησιμοποιήσαμε DNS κίνηση από τρεις διαφορετικές πηγές.

Ο μηχανισμός κατηγοριοποίησης και αντιμετώπισης που προτάθηκε ήταν ικανός να:

- επιλέγει με αυτοματοποιημένο τρόπο πεδία των πακέτων που απαιτούνται για την κατασκευή των signatures
- πετυχαίνει ακρίβειες κατηγοριοποίησης της τάξης του 99%, όσον αφορά τον εντοπισμό καλόβουλων και κακόβουλων πακέτων

- αντιμετωπίζει πιο σύντομα (χρονικά) αλλά και πιο αποδοτικά volumetric επιθέσεις σε σύγκριση με τους κλασικούς μηχανισμούς (που χρησιμοποιούν δικτυακές ροές).

Παρότι οι τεχνικές που βασίζονται σε signatures κατάφεραν να διακρίνουν με ακρίβεια την καλόβουλη από την κακόβουλη κίνηση προϋποθέτουν την ύπαρξη μεγάλου πλήθους από ετερογενή δεδομένα για την εκπαίδευση μοντέλων Επιβλεπόμενης Μάθησης. Όπως γίνεται αντιληπτό εντός ενός αυτόνομου δικτύου δεν είναι πάντα εφικτή η ύπαρξη ετερογενών δεδομένων με αποτέλεσμα την μειωμένη ακρίβεια ανίχνευσης. Επίσης, η αντιμετώπιση της επίθεσης (ακόμη και με τη χρήση signatures) από το δίκτυο θύμα δεν είναι πάντα εφικτή, όπως σε περιπτώσεις όπου οι επιθέσεις υπερκαλύπτουν την χωρητικότητα των γραμμών του. Οι δύο αυτοί λόγοι μας οδήγησαν στον σχεδιασμό ενός ολοκληρωμένου μηχανισμού συνεργατικής ανίχνευσης και αντιμετώπισης επιθέσεων. Στο **κεφάλαιο 8** παρουσιάζουμε τον συνεργατικό μηχανισμό προστασίας, ο οποίος επεκτείνει τους μηχανισμούς που παρουσιάστηκαν στα κεφάλαια 6 και 7.

Το Διαδίκτυο όπως λειτουργεί σήμερα αποτελεί μια συνεργασία μεταξύ αυτόνομων οντοτήτων (δικτύων), ωστόσο αυτή η συνεργασία δεν επεκτείνεται de facto σε μηχανισμούς προστασίας από επιθέσεις. Το βασικό πρόβλημα όσον αφορά τους μηχανισμούς ανίχνευσης είναι η ανταλλαγή (εν δυνάμει) απόρρητων δεδομένων, το οποίο είτε απαγορεύεται από κανονισμούς προστασίας δεδομένων, είτε αποφεύγεται λόγω της διστακτικότητας των διαχειριστών δικτύου. Εν αντιθέσει με τη συνεργατική ανίχνευση, η συνεργατική αντιμετώπιση επιθέσεων είναι περισσότερο διαδεδομένη. Ωστόσο, οι περισσότεροι μηχανισμοί που χρησιμοποιούνται βασίζονται σε κανόνες αποκοπής υλοποιημένους σε δικτυακές συσκευές και είτε αποκόπτουν όλη την κίνηση (τόσο την καλόβουλη όσο και την κακόβουλη) είτε χρησιμοποιούν δικτυακές ροές. Αυτές οι τακτικές εμφανίζουν περιορισμούς όσον αφορά την κλιμακώσιμότητα, την ευελιξία και την απόδοση τους.

Με γνώμονα τις προαναφερθείσες προκλήσεις, στο **κεφάλαιο 8** προτείνεται ένας μηχανισμός συνεργατικής ανίχνευσης και αντιμετώπισης που βασίζεται σε signatures. Συγκεκριμένα, ο μηχανισμός ανίχνευσης εκμεταλλεύεται τεχνικές Ομόσπονδης Μάθησης (*Federated Learning*), που επιτρέπουν την συνεργατική εκπαίδευση μοντέλων Μηχανικής Μάθησης χωρίς την ανταλλαγή δεδομένων αλλά με την ανταλλαγή βαρών Νευρωνικών Δικτύων (*Neural Networks*). Από την άλλη, ο μηχανισμός αντιμετώπισης

βασίζεται σε προγραμματιζόμενες κάρτες δικτύου (XDP) και δίνει τη δυνατότητα για αποκοπή κακόβουλης κίνησης τόσο για ιδία χρήση όσο και για αιτήματα από συνεργαζόμενους φορείς. Η προτεινόμενη λύση παρέχει κλιμακωσιμότητα για αύξηση των επιδόσεων κατ' απαίτηση, υψηλές αποδόσεις αλλά και ευελιξία στην αποκοπή της κίνησης.

Για την αξιολόγηση του μηχανισμού χρησιμοποιήθηκε testbed υψηλών ταχυτήτων αλλά και πραγματική δικτυακή κίνηση. Ειδικότερα χρησιμοποιήθηκαν δεδομένα καλόβουλης κίνησης από το ιαπωνικό δίκτυο WIDE τα οποία διαχωρίστηκαν καταλλήλως για την προσομοίωση διακριτών αυτόνομων δικτύων. Αντίστοιχα χρησιμοποιήθηκαν οι επιθέσεις από το σύνολο δεδομένων *Booters* που αναφέρθηκε στα προηγούμενα κεφάλαια. Στόχος του πειραματισμού ήταν η αξιολόγηση της συνεργατικής τεχνικής Ομόσπονδης Μάθησης σε πραγματικά δεδομένα και η σύγκριση της με τις ακρίβειες που θα πετύχαινε κάθε αυτόνομο δίκτυο μόνο του. Στα πλαίσια της συνεργατικής αντιμετώπισης, στόχος ήταν η αξιολόγηση της απόδοσης σε υψηλούς ρυθμούς πακέτων για πολλαπλά αιτήματα αποκοπής επιθέσεων υπό το πρίσμα της κλιμακωσιμότητας σε επίπεδο υπολογιστικών πυρήνων (CPU cores).

Από την πειραματική διαδικασία προέκυψαν τα παρακάτω συμπεράσματα:

- Η συνεργατική ανίχνευση επιθέσεων με τη χρήση της τεχνικής της Ομόσπονδης Μάθησης επέτρεψε σε συνεργαζόμενα δίκτυα να πετύχουν υψηλότερες ακρίβειες από αυτές που πετύχαιναν αν δεν συνεργαζόντουσαν, χωρίς όμως να εκθέτουν απόρρητα δεδομένα.
- Η υλοποίηση του μηχανισμού συνεργατικής αντιμετώπισης επιθέσεων κατάφερε να απορρίψει πολλαπλές επιθέσεις με αθροιστικά υψηλό ρυθμό πακέτων ανά δευτερόλεπτο και να κλιμακώσει κατ' απαίτηση τους υπολογιστικούς του πόρους.

Έπειτα από την ανασκόπηση των επιμέρους κεφαλαίων της διατριβής, μπορούμε να συνοψίσουμε παρακάτω τις κύριες συνεισφορές της:

<u>Άμεση και με ακρίβεια ανίχνευση επιθέσεων στο επίπεδο δεδομένων</u>: Ο μηχανισμός που προτάθηκε σε αυτή τη διατριβή χρησιμοποιεί τη γλώσσα P4 για να κατασκευάσει ένα σύστημα ανίχνευσης επιθέσεων στο επίπεδο δεδομένων. Συγκριτικά με τους κλασικούς μηχανισμούς που εκτελούνται στο επίπεδο ελέγχου, η προτεινόμενη λύση ανιχνεύει άμεσα και ακρίβεια επιθέσεις καθώς η δικτυακή κίνηση διέρχεται από

μεταγωγείς/δρομολογητές. Αυτό επιτρέπει την δημιουργία άμεσων αντίμετρων για την έγκαιρη καταστολή της επίθεσης.

Κατηγοριοποίηση ιδιαίτερων χαρακτηριστικών των δικτυακών πακέτων με τη χρήση ευφυών μηχανισμών τεχνητής νοημοσύνης: Οι κλασικοί μηχανισμοί προστασίας χρησιμοποιούν ως πηγή δεδομένων δικτυακές ροές. Η συγκεκριμένη τακτική εμφανίζει δυσκολίες όσον αφορά τη συλλογή, επεξεργασία αλλά ακόμα και αποθήκευση των δεδομένων αυτών και μπορεί να επιβραδύνει σημαντικά την ανίχνευση και την αντιμετώπιση επιθέσεων DDoS. Αντίθετα με αυτή τη στρατηγική, στη συγκεκριμένη διατριβή προτείναμε έναν μηχανισμό κατηγοριοποίησης που χρησιμοποιεί τα ιδιαίτερα χαρακτηριστικά των κακόβουλων πακέτων (signatures) για να ανιχνεύσει έγκαιρα επιθέσεις DDoS. Αυτά τα χαρακτηριστικά παράγονται με αυτοματοποιημένο τρόπο μέσα από τη χρήση ευφυών μοντέλων Μηχανικής Μάθησης που προσφέρουν δυνατότητες γενίκευσης της γνώσης που έχουν λάβει.

Αντιμετώπιση επιθέσεων με τη χρήση συνεκτικών κανόνων αποκοπής βάσει ιδιαίτερων χαρακτηριστικών των πακέτων: Οι κανόνες αποκοπής που χρησιμοποιούνται κατά κόρον για την αποσόβηση επιθέσεων DDoS εφαρμόζονται κατά κύριο λόγο σε δικτυακές συσκευές όπως τείχη προστασίας, δρομολογητές. Αυτές οι δικτυακές συσκευές εμφανίζουν περιορισμούς στο πλήθος των κανόνων που μπορούν να υποστηρίξουν αλλά και στους τύπους των κανόνων. Στη διατριβή αυτή παρουσιάσαμε ένα μηχανισμό κατασκευής κανόνων αποκοπής που δημιουργεί συνεκτικά σύνολα κανόνων που περιγράφουν με μεγάλη ακρίβεια τον τύπο της επίθεσης, χωρίς να επηρεάζονται σημαντικές ποσότητες της καλόβουλης κίνησης.

Υψηλής απόδοσης κλιμακώσιμες εικονικές δικτυακές λειτουργίες υλοποιημένες σε προγραμματιζόμενες υποδομές: Η παρακολούθηση αλλά και η αποκοπή της δικτυακής κίνησης σε legacy δικτυακά περιβάλλοντα υλοποιούνταν συνήθως από ειδικού τύπου εξοπλισμό. Στη διατριβή αυτή προτείναμε την υλοποίηση αυτών των δικτυακών λειτουργιών σε γενικού τύπου εξοπλισμό με τη χρήση του framework XDP. Με αυτό τον τρόπο δίνεται η δυνατότητα για σχεδιασμό κλιμακώσιμων και αποδοτικών εικονικών δικτυακών λειτουργιών κατάλληλες για τη χρήση σε σύγχρονα δικτυακά περιβάλλοντα υπολογιστικού νέφους.

Συνεργατική ανίχνευση και αντιμετώπιση επιθέσεων με γνώμονα την προστασία των προσωπικών δεδομένων: Το σημερινό διαδίκτυο είναι απόρροια συνεργασιών μεταξύ αυτόνομων δικτύων (Autonomous Systems). Η συνεργασία αυτή δεν επεκτείνεται όμως και για σκοπούς προστασίας των δικτύων από επιθέσεις. Η συνεργατική ανίχνευση περιορίζεται από διστακτικότητα αλλά και νόμους που αφορούν την ανταλλαγή προσωπικών δεδομένων. Από την άλλη η συνεργατική αντιμετώπιση δεν εφαρμόζεται σε μεγάλη κλίμακα λόγω εγγενών περιορισμών των δικτυακών συσκευών που εφαρμόζουν του κανόνες αποκοπής. Στην παρούσα διατριβή προτείναμε έναν μηχανισμό συνεργατικής ανίχνευσης και αντιμετώπισης που βασίζεται σε ιδιαίτερα χαρακτηριστικά των πακέτων. Η ανίχνευση των επιθέσεων γίνεται με μηχανισμούς Ομόσπονδης Μάθησης, μια τεχνική που δεν απαιτεί την ανταλλαγή απόρρητων δεδομένων ενώ η αντιμετώπιση με τη χρήση γενικού τύπου προγραμματιζόμενων συσκευών (XDP) που παρέχουν ευελιξία, κλιμακωσιμότητα και υψηλή απόδοση.

Υλοποίηση σε πραγματικό δικτυακό και υπολογιστικό εξοπλισμό και αξιολόγηση των μεθοδολογιών με τη χρήση πραγματικών δεδομένων: Στα πλαίσια της τρέχουσας διατριβής, οι προτεινόμενες υλοποιήσεις δοκιμάστηκαν σε testbed υψηλών ταχυτήτων με στόχο την αξιολόγηση τους σε ρεαλιστικές συνθήκες. Τα δεδομένα που χρησιμοποιήθηκαν για την αξιολόγηση των προτεινόμενων μεθόδων προέρχονται από πραγματικά ετερογενή δικτυακά περιβάλλοντα επιτρέποντας μας να δοκιμάσουμε τις προσεγγίσεις μας σε πραγματικές συνθήκες.

11 Publications

11.1 Articles in Scientific Journals

- **M. Dimolianis**, A. Pavlidis & V. Maglaris, "*Signature-based Traffic Classification and Mitigation for DDoS Attacks using Programmable Network Data Planes*", in *IEEE Access*, vol. 9, pp. 113061-113076, 2021
- H. Harkous, C. Papagianni, K.D. Schepper, M. Jarschel, **M. Dimolianis** & R. Preis, "*Virtual queues for p4: A poor man's programmable traffic manager*", in *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2860-2872, 2021
- A. Pavlidis, **M. Dimolianis**, K. Giotis, L. Anagnostou, N. Kostopoulos, T. Tsigkritis, I. Kotinas, D. Kalogeras & V. Maglaris "*Orchestrating DDoS Mitigation via Blockchain-based Network Provider Collaborations*", in *The Knowledge Engineering Review*, vol. 35, pp. 1–17, 2020
- P. Vuletić, B. Bosak, **M. Dimolianis**, P. Mérindol, D. Schmitz & H. Wessing,"*Localization of network service performance degradation in multi-tenant networks*", in *Computer Networks*, vol. 168, 2019

11.2 Papers in Conferences

- **M. Dimolianis**, A. Pavlidis & V. Maglaris, "*SYN Flood Attack Detection and Mitigation using Machine Learning Traffic Classification and Programmable Data Plane Filtering*", in Proceedings of the *Conference on Innovation in Clouds, Internet and Networks and Workshops*, 2021
- **M. Dimolianis**, A. Pavlidis &V. Maglaris, "*A Multi-Feature DDoS Detection Schema on P4 Hardware*", in Proceedings of the *Conference on Innovation in Clouds, Internet and Networks and Workshops*, 2020
- N. Kostopoulos, A. Pavlidis, **M. Dimolianis**, D. Kalogeras & V. Maglaris, "*A Privacy-Preserving Schema for the Detection and Collaborative Mitigation of DNS Water Torture Attacks in Cloud Infrastructures*", in Proceedings of the *International Conference on Cloud Networking*, 2019
- A. Pavlidis, **M. Dimolianis**, D. Kalogeras & V. Maglaris, "*Automated Distribution of Access Control Rules in Defense Layers of an Enterprise*

Network", in Proceedings of the *Symposium on Integrated Network and Service Management*, 2019

- **M. Dimolianis**, A. Pavlidis, D. Kalogeras & V. Maglaris, "*Mitigation of Multi-vector Network Attacks via Orchestration of Distributed Rule Placement*", in Proceedings of the *Symposium on Integrated Network and Service Management*, 2019
- K. Giotis, A. Pavlidis, L. Anagnostou, **M. Dimolianis**, T. Tsigkritis, D. Kalogeras, N. Kostopoulos, I. Kotinas & V. Maglaris, "*Blockchain-based Federation of Network Providers for Collaborative DDoS Mitigation*", in Proceedings of the *3rd Symposium on Distributed Ledger Technology*, 2018

12 References

[1] "Denial-of-service attack - Wikipedia," available at: https://en.wikipedia.org/wiki/Denial-of-service_attack [accessed February 18, 2022]

[2] J. J. Santanna *et al.*, "Booters - An analysis of DDoS-as-a-service attacks," in *Proceedings of the International Symposium on Integrated Network Management*, pp. 243–251, 2015, DOI: https://dx.doi.org/10.1109/INM.2015.7140298

[3] "Arbor DDoS Protection Solutions," available at: https://www.netscout.com/arbor-ddos [accessed February 15, 2022]

[4] "Imperva DDoS Protection 3-Second SLA," available at: https://www.imperva.com/products/ddos-protection-services/ [accessed February 15, 2022]

[5] "Cloudflare DDoS Protection & Mitigation," available at: https://www.cloudflare.com/ddos/ [accessed February 15, 2022]

[6] "F5 DDoS Hybrid Defender," available at: https://www.f5.com/products/security/ddos-hybrid-defender [accessed February 15, 2022]

[7] "A10 Networks DDoS Detection & Mitigation: Thunder TPS," available at: https://www.a10networks.com/products/thunder-tps/ [accessed February 18, 2022]

[8] "FortiDDoS DDoS Protection Solution," available at: https://www.fortinet.com/products/ddos/fortiddos [accessed February 18, 2022]

[9] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," in *Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008, DOI: https://dx.doi.org/10.1145/1355734.1355746

[10] M. Casado, M. J. Freedman, J. Pettit, J. Luo, N. McKeown, and S. Shenker, "Ethane: taking control of the enterprise," in *Computer Communication Review*, vol. 37, no. 4, pp. 1–12, 2007, DOI: https://dx.doi.org/10.1145/1282427.1282382

[11] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," in *Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014, DOI: https://dx.doi.org/10.1145/2656877.2656890

[12] T. Høiland-Jørgensen *et al.*, "The eXpress data path: Fast programmable packet

processing in the operating system kernel," in *Proceedings of the International Conference on Emerging Networking EXperiments and Technologies*, pp. 54–66, 2018, DOI: https://dx.doi.org/10.1145/3281411.3281443

[13] P. Phaal, S. Panchen, and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," available at: https://tools.ietf.org/html/rfc3176 [accessed March 17, 2022]

[14] B. Claise, "Cisco Systems NetFlow Services Export Version 9," available at: https://datatracker.ietf.org/doc/html/rfc3954 [accessed March 17, 2022]

[15] S. Kaur, K. Kumar, and N. Aggarwal, "A review on P4-Programmable data planes: Architecture, research efforts, and future directions," in *Computer Communications*, vol. 170, pp. 109–129, 2021, DOI: https://dx.doi.org/10.1016/j.comcom.2021.01.027

[16] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," in *Computer Networks*, vol. 62, pp. 122–136, 2014, DOI: https://dx.doi.org/10.1016/j.bjp.2013.10.014

[17] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proceedings of the Local Computer Network Conference*, pp. 408–415, 2010, DOI: https://dx.doi.org/10.1109/LCN.2010.5735752

[18] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined WAN," in *Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013, DOI: https://dx.doi.org/10.1145/2534169.2486019

[19] "Software-Defined Networking (SDN) Definition - Open Networking Foundation," available at: https://opennetworking.org/sdn-definition/ [accessed June 8, 2021]

[20] "Data Plane Development Kit - DPDK," available at: https://www.dpdk.org/ [accessed March 17, 2022]

[21] "Facebook, Google, Isovalent, Microsoft, and Netflix announce eBPF Foundation," available at: https://isovalent.com/blog/post/2021-08-ebpf-foundation-announcement [accessed September 7, 2021]

[22] A. Fabre, "L4Drop: XDP DDoS Mitigations," available at: https://blog.cloudflare.com/l4drop-xdp-ebpf-based-ddos-mitigations/ [accessed

October 14, 2020]

[23] “Katran: A high performance layer 4 load balancer,” available at: https://github.com/facebookincubator/katran [accessed March 17, 2022]

[24] “eBPF and XDP - Suricata,” available at: https://suricata.readthedocs.io/en/latest/capture-hardware/ebpf-xdp.html [accessed March 17, 2022]

[25] “In-Network Computing,” available at: https://www.sigarch.org/in-network-computing-draft/ [accessed September 9, 2021]

[26] “Tofino 2 - Barefoot,” available at: https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-2-series.html [accessed June 16, 2020]

[27] “Netronome Agilio SmartNICs,” available at: https://www.netronome.com/products/agilio-cx/ [accessed March 17, 2022]

[28] “P4-16 Language Specification,” available at: https://p4.org/p4-spec/docs/P4-16-v1.2.0.html [accessed March 17, 2022]

[29] “BEHAVIORAL MODEL (BMv2): The reference P4 software switch,” available at: https://github.com/p4lang/behavioral-model [accessed September 9, 2021]

[30] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, “Heavy-hitter detection entirely in the data plane,” in *Proceedings of the Symposium on SDN Research*, pp. 164–176, 2017, DOI: https://dx.doi.org/10.1145/3050220.3063772

[31] A. C. Lapolli, J. Adilson Marques, and L. P. Gaspary, “Offloading real-time DDoS attack detection to programmable data planes,” in *Proceedings of the Symposium on Integrated Network Management*, pp. 19–27, 2019

[32] M. Dimolianis, A. Pavlidis, and V. Maglaris, “A Multi-Feature DDoS Detection Schema on P4 Network Hardware,” in *Proceedings of the Conference on Innovation in Clouds, Internet and Networks and Workshops*, pp. 1–6, 2020, DOI: https://dx.doi.org/10.1109/ICIN48450.2020.9059327

[33] H. Harkous, C. Papagianni, K. De Schepper, M. Jarschel, M. Dimolianis, and R. Preis, “Virtual Queues for P4: A Poor Man’s Programmable Traffic Manager,” in *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 2860–2872, 2021, DOI: https://dx.doi.org/10.1109/TNSM.2021.3077051

[34] “Network Functions Virtualisation (NFV); Architectural Framework Group Specification - ETSI GS NFV 002 V1.1.1,” 2013, available at:

https://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf [accessed March 17, 2022]

[35] "PF_RING – ntop," available at: https://www.ntop.org/products/packet-capture/pf_ring/ [accessed March 17, 2022]

[36] L. Deri, "Using nDPI over DPDK to Classify and Block Unwanted Network Traffic Traffic Classification : an Overview," available at: https://www.dpdk.org/wp-content/uploads/sites/35/2018/12/LUCADERI_NDPI.pdf [accessed November 9, 2021]

[37] B. Perlman and E. Networks, "Accelerating Telco NFV Deployments with DPDK and SmartNICs," 2018, available at: https://www.dpdk.org/wp-content/uploads/sites/35/2018/12/Kalimani-and-Barak-Accelerating-NFV-with-DPDK-and-SmartNICs.pdf [accessed November 9, 2021]

[38] "Can SNMP (Still) Be Used to Detect DDoS Attacks?," available at: https://blog.radware.com/security/2018/08/snmp-ddos-attack-detection/ [accessed December 18, 2020]

[39] "SNMP is dead – Google Research NANOG 2018," available at: https://research.google/pubs/pub47773/ [accessed December 18, 2020]

[40] A. Douitsis and V. Maglaris, "Towards a scalable management collector," in *Proceedings of the Global Information Infrastructure and Networking Symposium*, pp. 1–6, 2016, DOI: https://dx.doi.org/10.1109/GIIS.2016.7814939

[41] "Google Protocol Buffers," available at: https://developers.google.com/protocol-buffers [accessed February 22, 2022]

[42] "Juniper - Streaming Telemetry," available at: https://www.juniper.net/documentation/en_US/junos/topics/reference/general/junos-telemetry-interface-grpc-sensors.html [accessed December 21, 2020]

[43] P. Vuletic, M. Dimolianis, V. Olifer, and I. Golub, "Zero-Footprint Monitoring - GÉANT White Paper," 2020, available at: https://about.geant.org/wp-content/uploads/2021/12/Zero-Footprint-Monitoring.pdf [accessed March 18, 2022]

[44] "NetFlow - Wikipedia," available at: https://en.wikipedia.org/wiki/NetFlow [accessed April 7, 2022]

[45] B. Rashidi, C. Fung, and E. Bertino, "A Collaborative DDoS Defence Framework Using Network Function Virtualization," in *IEEE Transactions on*

Information Forensics and Security, vol. 12, no. 10, pp. 2483–2497, 2017, DOI: https://dx.doi.org/10.1109/TIFS.2017.2708693

[46] "Snort - Network Intrusion Detection & Prevention System," available at: https://www.snort.org/ [accessed March 18, 2022]

[47] M. Prince, "The DDoS That Knocked Spamhaus Offline (And How We Mitigated It) - Cloudflare," available at: https://blog.cloudflare.com/the-ddos-that-knocked-spamhaus-offline-and-ho/ [accessed March 18, 2022]

[48] M. Majkowski, "Reflections on reflection (attacks)," available at: https://blog.cloudflare.com/reflections-on-reflections/ [accessed March 18, 2022]

[49] W. Eddy, "RFC 4987 - TCP SYN Flooding Attacks and Common Mitigations," available at: https://tools.ietf.org/html/rfc4987 [accessed March 18, 2022]

[50] "What Is an ACK Flood?," available at: https://www.cloudflare.com/learning/ddos/what-is-an-ack-flood/ [accessed January 11, 2021]

[51] "Anatomy of a SYN-ACK attack - Akamai Security Intelligence and Threat Research Blog," available at: https://blogs.akamai.com/sitr/2019/07/anatomy-of-a-syn-ack-attack.html [accessed January 11, 2021]

[52] "HTTP Flood DDoS Attack - Cloudflare," available at: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/ [accessed January 11, 2021]

[53] "Slowloris DDoS Attack - Cloudflare," available at: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/ [accessed January 11, 2021]

[54] "How to Defend DNS Services from All Types of DDoS Attacks - A10 Networks," available at: https://www.a10networks.com/blog/how-to-defend-dns-services-from-all-types-of-ddos-attacks/ [accessed January 11, 2021]

[55] S. Fichera, L. Galluccio, S. C. Grancagnolo, G. Morabito, and S. Palazzo, "OPERETTA: An OPEnflow-based REmedy to mitigate TCP SYNFLOOD Attacks against web servers," in *Computer Networks*, vol. 92, pp. 89–100, 2015, DOI: https://dx.doi.org/10.1016/j.comnet.2015.08.038

[56] P. Kumar, M. Tripathi, A. Nehra, M. Conti, and C. Lal, "SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN," in *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018, DOI: https://dx.doi.org/10.1109/TNSM.2018.2861741

[57] R. Mohammadi, R. Javidan, and M. Conti, “SLICOTS: An SDN-based lightweight countermeasure for TCP SYN flooding attacks,” in *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, pp. 487–497, 2017, DOI: https://dx.doi.org/10.1109/TNSM.2017.2701549

[58] K. Giotis, G. Androulidakis, and V. Maglaris, “A scalable anomaly detection and mitigation architecture for legacy networks via an OpenFlow middlebox,” in *Security and Communication Networks*, vol. 9, no. 13, pp. 422–437, 2015, DOI: https://dx.doi.org/10.1002/sec.1368

[59] J. Hill, M. Aloserij, and P. Grosso, “Tracking network flows with P4,” in *Proceedings of the Innovating the Network for Data-Intensive Science*, pp. 23–32, 2018, DOI: https://dx.doi.org/10.1109/INDIS.2018.00006

[60] “FastNetMon DDoS detection tool,” available at: https://fastnetmon.com/ [accessed September 5, 2020]

[61] C. Siaterlis and V. Maglaris, “Detecting incoming and outgoing DDoS attacks at the edge using a single set of network characteristics,” in *Proceedings of the Symposium on Computers and Communications*, pp. 469–475, 2005, DOI: https://dx.doi.org/10.1109/ISCC.2005.50

[62] Y. Cui *et al.*, “SD-Anti-DDoS: Fast and efficient DDoS defense in software-defined networks,” in *Journal of Network and Computer Applications*, vol. 68, pp. 65–79, 2016, DOI: https://dx.doi.org/10.1016/j.jnca.2016.04.005

[63] Q. Niyaz, W. Sun, and A. Y. Javaid, “A Deep Learning Based DDoS Detection System in Software-Defined Networking (SDN),” in *ICST Transactions on Security and Safety*, vol. 4, no. 12, 2017, DOI: https://dx.doi.org/10.4108/eai.28-12-2017.153515

[64] A. Santos da Silva, J. A. Wickboldt, L. Z. Granville, and A. Schaeffer-Filho, “ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN,” in *Proceedings of the IEEE/IFIP Network Operations and Management Symposium*, pp. 27–35, 2016, DOI: https://dx.doi.org/10.1109/NOMS.2016.7502793

[65] X. Yuan, C. Li, and X. Li, “DeepDefense: Identifying DDoS Attack via Deep Learning,” in *Proceedings of the International Conference on Smart Computing*, pp. 1–8, 2017, DOI: https://dx.doi.org/10.1109/SMARTCOMP.2017.7946998

[66] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del-Rincon, and D. Siracusa, “LUCID: A Practical, Lightweight Deep Learning Solution for DDoS

Attack Detection," in *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020, DOI: https://dx.doi.org/10.1109/TNSM.2020.2971776

[67] "Zero-Day Exploits & Zero-Day Attacks," available at: https://www.kaspersky.com/resource-center/definitions/zero-day-exploit [accessed February 24, 2022]

[68] C. Dietzel, A. Feldmann, and T. King, "Blackholing at IXPs: On the Effectiveness of DDoS Mitigation in the Wild," in *Proceedings of the International Conference on Passive and Active Network Measurement*, pp. 319–332, 2016, DOI: https://dx.doi.org/10.1007/978-3-319-30505-9_24

[69] L. Serodio, "Traffic Diversion Techniques for DDoS Mitigation using BGP Flowspec Distributed Denial of Service (DDoS) Attacks," available at: https://archive.nanog.org/sites/default/files/wed.general.trafficdiversion.serodio.10.pdf [accessed March 18, 2022]

[70] W. Kumari and D. McPherson, "RFC 5635 - Remote Triggered Black Hole Filtering with Unicast Reverse Path Forwarding (uRPF)," available at: https://datatracker.ietf.org/doc/html/rfc5635 [accessed March 18, 2022]

[71] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," in *Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004, DOI: https://dx.doi.org/10.1145/997150.997156

[72] F. Soldo, K. Argyraki, and A. Markopoulou, "Optimal source-based filtering of malicious traffic," in *IEEE/ACM Transactions on Networking*, vol. 20, no. 2, pp. 381–395, 2012, DOI: https://dx.doi.org/10.1109/TNET.2011.2161615

[73] P. Marques, N. Sheth, R. Raszuk, B. Greene, J. Mauch, and P. McPherson, "RFC 5575 - Dissemination of Flow Specification Rules," available at: https://datatracker.ietf.org/doc/html/rfc5575 [accessed March 18, 2022]

[74] P. Phaal, S. Panchen, and N. McKee, "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," Rfc 3176, 2001, available at: https://tools.ietf.org/html/rfc3176 [accessed March 17, 2022]

[75] "P4-16 Language Specification version 1.2.1," available at: https://p4.org/p4-spec/docs/P4-16-v1.2.1.html [accessed March 18, 2022]

[76] H. Harkous, M. Jarschel, M. He, R. Priest, and W. Kellerer, "Towards Understanding the Performance of P4 Programmable Hardware," in *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and*

Communications Systems, pp. 1–6, 2019, DOI: https://dx.doi.org/10.1109/ANCS.2019.8901881

[77] O. Hohlfeld, J. Krude, J. H. Reelfs, J. Ruth, and K. Wehrle, "Demystifying the Performance of XDP BPF," in *Proceedings of the Conference on Network Softwarization*, pp. 208–212, 2019, DOI: https://dx.doi.org/10.1109/NETSOFT.2019.8806651

[78] G. Bertin, "XDP in practice: integrating XDP in our DDoS mitigation pipeline," available at: https://legacy.netdevconf.info/2.1/papers/Gilberto_Bertin_XDP_in_practice.pdf [accessed March 18, 2022]

[79] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," in *Internet Mathematics*, vol. 1, no. 4, pp. 485–509, 2004, DOI: https://dx.doi.org/10.1080/15427951.2004.10129096

[80] K. Cho, K. Mitsuya, and A. Kato, "Traffic data repository at the WIDE project," in *Proceedings of the Annual Conference on USENIX Annual Technical Conference*, 2000

[81] "Monitoring Tools - GRNET," available at: https://mon.grnet.gr/ [accessed May 16, 2020]

[82] "Cloudflare - Network-layer DDoS attack trends for Q2 2020," available at: https://blog.cloudflare.com/network-layer-ddos-attack-trends-for-q2-2020/ [accessed October 21, 2020]

[83] "Imperva mitigates largest DDoS attacks of 2020 so far," available at: https://www.imperva.com/blog/imperva-mitigates-largest-ddos-attacks-of-2020-so-far/ [accessed October 21, 2020]

[84] D. Bernstein, "SYN cookies," available at: http://cr.yp.to/syncookies.html [accessed March 18, 2022]

[85] M. Majkowski, "SYN packet handling in the wild," available at: https://blog.cloudflare.com/syn-packet-handling-in-the-wild/ [accessed June 26, 2020]

[86] "Fortinet - Configuring a TCP SYN flood protection policy," available at: https://docs.fortinet.com/document/fortiadc/5.4.0/handbook/832593/configuring-a-tcp-syn-flood-protection-policy [accessed October 14, 2020]

[87] D. Scholz, S. Gallenmueller, H. Stubbe, B. Jaber, M. Rouhi, and G. Carle, "Me Love (SYN-) Cookies : SYN Flood Mitigation in Programmable Data Planes,"

in *arXiv preprint*, 2020, DOI: https://dx.doi.org/10.48550/arXiv.2003.03221

[88] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002, DOI: https://dx.doi.org/10.1109/4235.996017

[89] "TCP SYN Flood mitigation via XDP," available at: https://github.com/doup123/syn_flood_xdp [accessed October 22, 2020]

[90] "Platypus - Multiobjective Optimization in Python," available at: https://platypus.readthedocs.io/en/latest/ [accessed October 14, 2020]

[91] "RandomForestClassifier - scikit-learn," available at: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [accessed November 1, 2020]

[92] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations*, pp. 1–15, 2015

[93] G. Bertin, "Introducing the p0f BPF compiler," available at: https://blog.cloudflare.com/introducing-the-p0f-bpf-compiler/ [accessed October 23, 2020]

[94] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *in Proceedings of the Network and Distributed System Security Symposium*, 2014, DOI: https://dx.doi.org/10.14722/ndss.2014.23233

[95] "Global DDoS Summary - NETSCOUT Cyber Threat Horizon," available at: https://horizon.netscout.com/?atlas=summary [accessed September 17, 2020]

[96] S. Miano, R. Doriguzzi-Corin, F. Risso, D. Siracusa, and R. Sommese, "Introducing SmartNICs in Server-Based Data Plane Processing: The DDoS Mitigation Use Case," in *IEEE Access*, vol. 7, pp. 107161–107170, 2019, DOI: https://dx.doi.org/10.1109/access.2019.2933491

[97] "Suricata Open Source IDS/IPS," available at: https://suricata.io/ [accessed March 18, 2022]

[98] Y. Afek, A. Bremler-Barr, and S. L. Feibish, "Zero-Day Signature Extraction for High-Volume Attacks," in *IEEE/ACM Transactions on Networking*, vol. 27, no. 2, pp. 691–706, 2019, DOI: https://dx.doi.org/10.1109/TNET.2019.2899124

[99] "What is Out of Bag (OOB) score in Random Forest?," available at: https://towardsdatascience.com/what-is-out-of-bag-oob-score-in-random-forest-

a7fa23d710 [accessed April 21, 2021]

[100] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot, “Variable selection using random forests,” in *Pattern Recognition Letters*, vol. 31, no. 14, pp. 2225–2236, 2010, DOI: https://dx.doi.org/10.1016/j.patrec.2010.03.014

[101] M. M. Singh, M. M. Singh, and S. Kaur, “10 Days DNS Network Traffic from April-May 2016,” available at: https://data.mendeley.com/datasets/zh3wnddzxy/1 [accessed March 18, 2022]

[102] “sFlow: Sampling rates,” available at: https://blog.sflow.com/2009/06/sampling-rates.html [accessed March 18, 2022]

[103] “Netronome Flow Processor (NFP) Kernel Drivers,” available at: https://www.kernel.org/doc/html/latest/networking/device_drivers/ethernet/netronome/nfp.html [accessed March 18, 2022]

[104] “General Data Protection Regulation - GDPR,” available at: https://gdpr-info.eu/ [accessed January 7, 2022]

[105] H. Brendan McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, “Communication-efficient learning of deep networks from decentralized data,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 1273–1282, 2017

[106] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, “Federated learning of predictive models from federated Electronic Health Records,” in *International Journal of Medical Informatics*, vol. 112, pp. 59–67, 2018, DOI: https://dx.doi.org/10.1016/j.ijmedinf.2018.01.007

[107] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, “A performance evaluation of federated learning algorithms,” in *Proceedings of the Workshop on Distributed Infrastructures for Deep Learning*, pp. 1–8, 2018, DOI: https://dx.doi.org/10.1145/3286490.3286559

[108] J. Li, L. Lyu, X. Liu, X. Zhang, and X. Lyu, “FLEAM: A Federated Learning Empowered Architecture to Mitigate DDoS in Industrial IoT,” in *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4059–4068, 2022, DOI: https://dx.doi.org/10.1109/TII.2021.3088938

[109] Q. Tian, C. Guang, C. Wenchao, and W. Si, “A lightweight residual networks framework for DDoS attack classification based on federated learning,” in *Proceedings of the Conference on Computer Communications Workshops*, 2021, DOI: https://dx.doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484622

[110] Y. Chen, K. Hwang, and W.-S. Ku, “Collaborative Detection of DDoS Attacks over Multiple Network Domains,” in *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 12, pp. 1649–1662, 2007, DOI: https://dx.doi.org/10.1109/TPDS.2007.1111

[111] J. François, I. Aib, and R. Boutaba, “FireCol: A collaborative protection network for the detection of flooding DDoS attacks,” in *IEEE/ACM Transactions on Networking*, vol. 20, no. 6, pp. 1828–1841, 2012, DOI: https://dx.doi.org/10.1109/TNET.2012.2194508

[112] J. Steinberger, B. Kuhnert, A. Sperotto, H. Baier, and A. Pras, “Collaborative DDoS defense using flow-based security event information,” in *Proceedings of the Network Operations and Management Symposium*, pp. 516–522, 2016, DOI: https://dx.doi.org/10.1109/NOMS.2016.7502852

[113] C. Hesselman and R. Yazdani, “CONCORDIA Cyber security cOmpeteNCe fOr Research anD InnovAtion DDoS Clearing House for Europe Cross-sector Pilot Demo,” available at: https://www.sidnlabs.nl/downloads/2deJudioEsd0oFWufTXdV9/099fa8c92f7d601e0669bec73b2fa272/NEW-20200123-CONCORDIA-T3.2-demo-review-final.pdf [accessed December 8, 2021]

[114] K. Giotis, M. Apostolaki, and V. Maglaris, “A reputation-based collaborative schema for the mitigation of distributed attacks in SDN domains,” in *Proceedings of the Network Operations and Management Symposium*, pp. 495–501, 2016, DOI: https://dx.doi.org/10.1109/NOMS.2016.7502849

[115] A. Pavlidis *et al.*, “Orchestrating DDoS mitigation via blockchain-based network provider collaborations,” in *Knowledge Engineering Review*, vol. 35, pp. 1–17, 2020, DOI: https://dx.doi.org/10.1017/S0269888920000259

[116] “The Global Leaders’ Forum launches Communications Blockchain Network (CBN) - Deutsche Telekom Global Carrier,” available at: https://globalcarrier.telekom.com/newsroom/news/news-pages/global-leaders-forum-launches-communications-blockchain-network [accessed January 13, 2022]

[117] Y. Zhao, J. Chen, D. Wu, J. Teng, and S. Yu, “Multi-Task Network Anomaly Detection using Federated Learning,” in *in Proceedings of the International Symposium on Information and Communication Technology*, pp. 273–279, 2019, DOI: https://dx.doi.org/10.1145/3368926.3369705

[118] “DE-CIX – Deutscher Commercial Internet Exchange,” available at: https://www.de-cix.net/ [accessed January 11, 2022]

[119] M. Dimolianis, A. Pavlidis, and V. Maglaris, “Signature-Based Traffic Classification and Mitigation for DDoS Attacks Using Programmable Network Data Planes,” in *IEEE Access*, vol. 9, pp. 113061–113076, 2021, DOI: https://dx.doi.org/10.1109/ACCESS.2021.3104115

[120] “Cisco ASR 9000 Series Aggregation Services Router - Implementing BGP Flowspec,” available at: https://www.cisco.com/c/en/us/td/docs/routers/asr9000/software/asr9k_r5-2/routing/configuration/guide/b_routing_cg52xasr9k/b_routing_cg52xasr9k_chapter_011.html [accessed January 7, 2022]

[121] “Jinja Documentation,” available at: https://jinja.palletsprojects.com/en/3.0.x/ [accessed December 2, 2021]

[122] A. Narayanan, S. Previdi, and F. Brian, “BGP Advertisements for Content URIs,” available at: https://slideplayer.com/slide/8149985/ [accessed December 13, 2021]

[123] “RabbitMQ message broker,” available at: https://www.rabbitmq.com/ [accessed December 9, 2021]

[124] “Ryu component-based software defined networking framework,” available at: https://github.com/faucetsdn/ryu [accessed January 7, 2022]

[125] “BGP Routing Table Analysis - IPv4 prefixes and their origin ASNs,” available at: https://thyme.apnic.net/current/ [accessed December 2, 2021]

[126] “Hyperparameter optimization - Grid search,” available at: https://en.wikipedia.org/wiki/Hyperparameter_optimization#Grid_search [accessed January 12, 2022]

[127] “perf (Linux) - Wikipedia,” available at: https://en.wikipedia.org/wiki/Perf_(Linux) [accessed January 5, 2022]

[128] M. Nawrocki, J. Blendin, C. Dietzel, T. C. Schmidt, and M. Wählisch, “Down the black hole: Dismantling operational practices of BGP blackholing at IXPS,” in *Proceedings of the Internet Measurement Conference*, pp. 435–448, 2019, DOI: https://dx.doi.org/10.1145/3355369.3355593

[129] Z. Liu *et al.*, “Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric DDoS attacks with programmable switches,” in *Proceedings of the USENIX Security Symposium*, pp. 3829–3846, 2021

[130] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” in *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, DOI: https://dx.doi.org/10.1162/neco.1997.9.8.1735

[131] J. Hawkins and S. Blakeslee, *On Intelligence: How a New Understanding of the Brain will Lead to the Creation of Truly Intelligent Machines*. Times Books, 2004

[132] Y. Feng, J. Li, and T. Nguyen, “Application-Layer DDoS Defense with Reinforcement Learning,” in *Proceedings of the International Symposium on Quality of Service*, 2020, DOI: https://dx.doi.org/10.1109/IWQoS49365.2020.9213026

[133] A. Gholami, N. Torkzaban, and J. S. Baras, “On the Importance of Trust in Next-Generation Networked CPS Systems: An AI Perspective,” in *arXiv preprint*, 2021, DOI: https://dx.doi.org/10.48550/arXiv.2104.07853