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 Countering Domain Name System (DNS) Cyber Attacks in Software-Defined Network Infrastructures

Doctoral Dissertation

of

Nikolaos G. Kostopoulos

Supervisor: Vasilis Maglaris, Professor Emeritus, NTUA

Athens, February 2024

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΔΙΑΧΕΙΡΙΣΗΣ ΚΑΙ ΒΕΛΤΙΣΤΟΥ ΣΧΕΔΙΑΣΜΟΥ ΔΙΚΤΥΩΝ ΤΗΛΕΜΑΤΙΚΗΣ

'Εξυπνες Υπηρεσίες Αντιμετώπισης Κυβερνοεπιθέσεων στο Σύστημα Ονοματοδοσίας (DNS) σε Προγραμματιζόμενες Διαδικτυακές Υποδομές

Διδακτορική Διατριβή

του

Νικολάου Γ. Κωστόπουλου

Συμβουλευτική Επιτροπή: Βασίλειος Μάγκλαρης, Ομ. Καθηγητής ΕΜΠ (Επιβλέπων)
Ευστάθιος Συκάς, Ομ. Καθηγητής ΕΜΠ
Νεκτάριος Κοζύρης, Καθηγητής ΕΜΠ

Εγκρίθηκε από την επταμελή εξεταστική επιτροπή την 14η Φεβρουαρίου 2024

. . . . .	. . . . .	. . . . .
Βασίλειος Μάγκλαρης	Ευστάθιος Συκάς	Νεκτάριος Κοζύρης
Ομ. Καθηγητής ΕΜΠ	Ομ. Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ
. . . . .	. . . . .	. . . . .
Δημήτριος Σούντρης	Συμεών Παπαβασιλείου	Γεώργιος Στάμου
Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ	Καθηγητής ΕΜΠ

. . . . . .
John Baras
Professor University of Maryland

Αθήνα, Φεβρουάριος 2024

.....

Νικόλαος Γ. Κωστόπουλος
Διδάκτωρ Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

To my Mom, Yota

Abstract

Domain Name System (DNS) maps symbolic identifiers (i.e. domain names) to various types of data, mainly IP addresses. Most network services depend on domain names for their operation, thus DNS is vital for the reliability of computer networks. This makes DNS a frequent target of Distributed Denial of Service (DDoS) attacks. One of the most devastating attack vectors against the DNS infrastructure is Water Torture, which floods Authoritative DNS Servers with invalid DNS requests to disrupt their name resolution services. Apart from DDoS attacks, DNS may also be employed to forward data pertaining to botnet activities. Domain Generation Algorithms (DGA's) abuse DNS to establish communications between compromised hosts (bots) and botnet orchestrators.

This dissertation addresses mechanisms for effectively protecting against threats targeting DNS infrastructures (i.e. Water Torture) or exploiting DNS messages for malicious purposes (i.e. DGA's). We overcome the limitations of traditional DNS security systems by employing Big Data methods (probabilistic data structures, Machine Learning - ML) and promising networking technologies (Software-Defined Networking, data plane programmability).

We initially propose a user-space schema that effectively detects and mitigates Water Torture attacks against Authoritative DNS Servers. Instead of processing plaintext names, we employ Bloom Filters (BF's) to map entire zone contents and filter DNS traffic in a time and space efficient manner. Our proposed mechanism categorizes requester IP addresses as legitimate or suspicious based on operations performed by probabilistic data structures (BF's, Count-Min Sketches) and deterministic spelling correction algorithms (SymSpell). DNS traffic from suspicious IP addresses is subsequently filtered based on BF contents. Malicious DNS requests are dropped, whereas benign requests are forwarded to the victim Authoritative DNS Server.

We subsequently employ data plane programmability to significantly improve the Water Torture attack filtering throughput of our user-space, BF-based mitigation mechanism. We rely on eXpress Data Path (XDP) to efficiently perform DNS Deep Packet Inspection (DPI) and rapidly discern benign and malicious requests at the Authoritative DNS server data plane. Contrary to other data plane programming approaches that require specialized hardware (e.g. P4 switches) or bypass the Linux kernel (e.g. Data Plane Development Kit - DPDK), our XDP-based approach runs on low-cost Network Interface Cards (NIC's) and does not bypass the kernel, thus Linux utilities (e.g. TCP/IP libraries) are available to developers.

We then propose a privacy-aware mechanism to facilitate Authoritative DNS Server zone distribution to external mitigation systems operating at Recursive DNS Servers (Recursors) or upstream scrubbing facilities. Without requiring formal agreements, our mechanism may enable administrators to filter DNS DDoS attacks more efficiently closer to their origins or DGA traffic with higher accuracy. Zones are mapped within Cuckoo Filters due to their time, space and dynamic element update advantages over BF's.

The aforementioned mechanisms depend on zone contents. However, Authoritative DNS Server administrators may be unwilling to collaborate due to security concerns. To that end, we complement our zone-based approaches for Water Torture attack mitigation by employing XDP to accelerate ML feature extraction and inference within the data plane of Recursors. Our approach relies on Naive Bayes Classifiers, which effectively segregate benign from DDoS attack DNS requests entirely within the Linux kernel. Thus, DNS attack mitigation throughput is significantly improved without requirements for any specialized hardware, e.g. Graphics Processing Units (GPU's).

Finally, we employ eXplainable Artificial Intelligence (XAI) techniques, specifically SHapley Additive exPlanation (SHAP), to interpet the decisions of binary ML classifiers that differentiate between legitimate names and malicious ones produced by DGA's. SHAP operates in a

model-agnostic, post-hoc manner, i.e. regardless of the utilized ML algorithm and after the completion of the learning phase; this enables the analysis of tree and deep neural network classifiers in a unified fashion. Based on various SHAP visualization tools (summary, dependence, force plots), we derive global and local model interpretations on multiple and single dataset sample points respectively; these enable us to estimate feature importance and assess feature interactions. Our approach relies on features that capture name statistical and linguistic properties, thus time-consuming and privacy sensitive operations to obtain historical data are avoided. Our evaluation focuses on determining how name classification decisions are affected by specific DGA schemes (i.e. arithmetic, wordlist, hash and permutation based) used for name construction.

Our proposed DNS protection mechanisms are evaluated via proof-of-concept setups within our laboratory virtualized infrastructure based on datasets widely-employed by the research community and synthetic traffic generated based on our experience from the National Technical University of Athens (NTUA) campus network services.

***Keywords*—** Domain Name System (DNS), Distributed Denial of Service (DDoS) Attacks, Water Torture, Domain Generation Algorithm (DGA), Software-Defined Network (SDN), Data Plane Programmability, eXpress Data Path (XDP), Machine Learning (ML), Privacy-Aware Mechanisms

Περίληψη

Το Σύστημα Ονοματοδοσίας (Domain Name System, DNS) απεικονίζει συμβολικά αναγνωριστικά (ονόματα ή domain names) σε διάφορους τύπους δεδομένων, κυρίως διευθύνσεις πρωτοκόλλου Διαδικτύου (Internet Protocol, IP). Οι περισσότερες δικτυακές υπηρεσίες βασίζονται σε ονόματα για τη λειτουργία τους. Κατά συνέπεια, το DNS είναι ζωτικής σημασίας για την αξιοπιστία των δικτύων υπολογιστών. Αυτό καθιστά το DNS συχνό στόχο κατανεμημένων επιθέσεων άρνησης παροχής υπηρεσίας (Distributed Denial of Service attacks ή επιθέσεις DDoS). Μία από τις πιο καταστροφικές επιθέσεις εναντίον υποδομών DNS είναι η Water Torture, η οποία πλημμυρίζει επίσημους εξυπηρετητές DNS (Authoritative DNS Servers) με μη έγκυρα ερωτήματα DNS για να διακόψει τις υπηρεσίες επίλυσης ονομάτων (name resolution). Εκτός από τις επιθέσεις DDoS, το DNS μπορεί επίσης να χρησιμοποιηθεί για την προώθηση δεδομένων που σχετίζονται με ενέργειες δικτύων μολυσμένων συσκευών (botnets). Οι αλγόριθμοι παραγωγής ονομάτων (Domain Generation Algorithms, DGA's) καταχρώνται το DNS για να υποστηρίξουν την επικοινωνία ανάμεσα σε μολυσμένους υπολογιστές (bots) και τους ενορχηστρωτές των botnets.

Η διατριβή αυτή προτείνει μηχανισμούς για την αποτελεσματική προστασία από απειλές που στοχεύουν υποδομές DNS (Water Torture) ή εκμεταλλεύονται μηνύματα DNS για κακόβουλους σκοπούς (DGA's). Οι περιορισμοί των καθιερωμένων συστημάτων ασφαλείας DNS ξεπερνώνται με μεθόδους μεγάλων δεδομένων (Big Data), δηλαδή πιθανοτικές δομές δεδομένων και αλγορίθμους μηχανικής μάθησης (Machine Learning, ML), καθώς και πολλά υποσχόμενες δικτυακές τεχνολογίες, δηλαδή δίκτυα που καθορίζονται από λογισμικό (Software-Defined Networks, SDN's) και προγραμματισμό σε επίπεδο δεδομένων (data plane programmability).

Αρχικά, προτείνουμε έναν μηχανισμό σε χώρο χρήστη (user space) που ανιχνεύει και αντιμετωπίζει αποδοτικά επιθέσεις Water Torture σε Authoritative DNS Servers. Αντί να επεξεργαζόμαστε ονόματα στην αυθεντική τους μορφή (plaintext), βασιζόμαστε στα Bloom Filters (BF's) για να απεικονίσουμε τα περιεχόμενα ολόκληρων ζωνών και να φιλτράρουμε επιθετική κίνηση DNS με τρόπο αποδοτικό ως προς τον χρόνο και τη διαθέσιμη μνήμη. Ο προτεινόμενος μηχανισμός κατηγοριοποιεί τις διευθύνσεις IP των αιτούντων ως καλόβουλες ή ύποπτες βάσει ενεργειών που εκτελούνται από πιθανοτικές δομές δεδομένων (BF's, Count-Min Sketches) και αιτιοκρατικούς (deterministic) αλγορίθμους διόρθωσης τυπογραφικών λαθών (SymSpell). Στη συνέχεια, η κίνηση DNS από ύποπτες διευθύνσεις IP φιλτράρεται βάσει των περιεχομένων των BF's. Τα κακόβουλα ερωτήματα DNS απορρίπτονται, ενώ τα καλόβουλα ερωτήματα προωθούνται στον Authoritative DNS Server που δέχεται την επίθεση.

Στη συνέχεια, χρησιμοποιούμε προγραμματισμό σε επίπεδο δεδομένων για να βελτιώσουμε σημαντικά την απόδοση φιλταρίσματος του, βασισμένου σε BF's, user-space μηχανισμού αντιμετώπισης επιθέσεων Water Torture που προτάθηκε προηγουμένως. Στηριζόμαστε στο eXpress Data Path (XDP) για την αποτελεσματική εκτέλεση ενεργειών DNS που βασίζονται σε βαθιά επιθεώρηση πακέτου (Deep Packet Inspection, DPI) με σκοπό την ταχύτατη διάκριση καλόβουλων και κακόβουλων ερωτημάτων DNS στο επίπεδο δεδομένων των Authoritative DNS Servers. Σε αντίθεση με άλλες μεθόδους προγραμματισμού σε επίπεδο δεδομένων, η προσέγγισή μας βασίζεται στο XDP και, κατά συνέπεια, δεν απαιτείται εξειδικευμένο υλικό (όπως στο P4) και δεν παρακάμπτεται ο πυρήνας του Linux (όπως στο Data Plane Development Kit- DPDK). Έτσι, ο προτεινόμενος μηχανισμός μπορεί να εφαρμοστεί σε χαμηλού κόστους κάρτες δικτύων (Network Interface Cards, NIC's), ενώ χρήσιμα εργαλεία του πυρήνα του Linux (π.χ. βιβλιοθήκες TCP/IP) παραμένουν διαθέσιμα στους προγραμματιστές.

Έπειτα προτείνουμε έναν privacy-aware μηχανισμό διανομής ζωνών από Authoritative DNS Servers σε εξωτερικά συστήματα άμυνας. Τέτοια συστήματα μπορεί να λειτουργούν σε αναδρομικούς εξυπηρετητές DNS (Recursive DNS Servers ή Recursors) ή σε υπηρεσίες που φιλτράρουν δικτυακή κίνηση (upstream scrubbing services). Χωρίς να απαιτούνται επίσημες συμφωνίες, ο μηχανισμός μας επιτρέπει σε διαχειριστές δικτύων να αντιμετωπίσουν επιθέσεις DDoS, που βασίζονται στο DNS, πιο αποτελεσματικά κοντά στις πηγές τους ή να φιλτράρουν κίνηση που παράγεται από DGA's με μεγαλύτερη ακρίβεια. Οι ζώνες απεικονίζονται σε Cuckoo Filters, καθώς πραγματοποιούν ταχύτερους ελέγχους ονομάτων και καταναλώνουν λιγότερη μνήμη σε σχέση με τα BF's, ενώ υποστηρίζουν δυναμικές ανανεώσεις ονομάτων.

Οι προαναφερθέντες μηχανισμοί εξαρτώνται από τα περιεχόμενα των ζωνών. Ωστόσο, οι διαχειριστές των Authoritative DNS Servers μπορεί να είναι απρόθυμοι να συνεργαστούν λόγων ανησυχιών για την ασφάλεια των υποδομών τους. Για το σκοπό αυτό, συμπληρώνουμε τις προηγούμενες προσεγγίσεις μας που προτάθηκαν για την αντιμετώπιση επιθέσεων Water Torture, χρησιμοποιώντας XDP για την επιτάχυνση της εξαγωγής χαρακτηριστικών (features) και την αποτίμηση των εξόδων αλγορίθμων ML στο επίπεδο δεδομένων των Recursors. Η προσέγγισή μας βασίζεται σε ταξινομητές Naive Bayes, οι οποίοι διαχωρίζουν αποτελεσματικά την καλόβουλη από την κακόβουλη κίνηση DNS εξ ολοκλήρου εντός του πυρήνα του Linux. Έτσι, η απόδοση αντιμετώπισης επιθέσεων DNS βελτιώνεται σημαντικά χωρίς να απαιτείται εξειδικευμένο υλικό, π.χ. κάρτες γραφικών.

Τέλος, εφαρμόζουμε τεχνικές επεξηγήσιμης τεχνητής νοημοσύνης (eXplainable Artificial Intelligence, XAI), συγκεκριμένα την SHapley Additive exPlanation (SHAP), για να ερμηνεύσουμε τις αποφάσεις δυαδικών ταξινομητών μηχανικής μάθησης που διαχωρίζουν καλόβουλα ονόματα από κακόβουλα, τα οποία παράγονται από DGA's. Η SHAP εκτελείται με model-agnostic, post-hoc τρόπο, δηλαδή ανεξάρτητα από τον χρησιμοποιούμενο αλγόριθμο ML και αφού ολοκληρωθεί η φάση μάθησης. Αυτό επιτρέπει την ανάλυση ταξινομητών που βασίζονται σε δέντρα αποφάσεων και βαθιά νευρωνικά δίκτυα με ενιαίο τρόπο. Βασισμένοι σε διάφορα εργαλεία οπτικοποίησης της SHAP (διαγράμματα summary, dependence και force), εξάγουμε ερμηνείες μοντέλων σε πολλαπλά (καθολικές ερμηνείες) και μονωμένα (τοπικές ερμηνείες) παραδείγματα του συνόλου δεδομένων. Έτσι, εκτιμούμε τη συνεισφορά των features και αξιολογούμε τις μεταξύ τους αλληλεπιδράσεις. Η προσέγγισή μας βασίζεται σε features τα οποία αντικατοπτρίζουν τις στατιστικές και γλωσσικές ιδιότητες των ονομάτων. Κατά συνέπεια, αποφεύγονται χρονοβόρες ενέργειες που βασίζονται σε ιστορικά δεδομένα και ενδέχεται να εγείρουν ζητήματα ιδιωτικότητας. Η αξιολόγησή μας επικεντρώνεται στον προσδιορισμό του τρόπου με τον οποίο επηρεάζονται οι αποφάσεις ταξινόμησης ονομάτων από συγκεκριμένους μηχανισμούς παραγωγής ονομάτων DGA (arithmetic, wordlist, hash, permutation based).

Οι προτεινόμενοι μηχανισμοί προστασίας DNS αξιολογούνται μέσω πειραμάτων στην εικονική υποδομή του εργαστηρίου μας. Τα πειράματα βασίζονται σε σύνολα δεδομένων που χρησιμοποιούνται ευρέως από την ερευνητική κοινότητα, καθώς και σε συνθετική δικτυακή κίνηση που κατασκευάσαμε βασισμένοι στην εμπειρία μας από τις δικτυακές υπηρεσίες του Εθνικού Μετσοβίου Πολυτεχνείου (ΕΜΠ).

Keywords--- Σύστημα Ονοματοδοσίας, Κατανεμημένες επιθέσεις άρνησης παροχής υπηρεσίας, Επιθέσεις Water Torture, Αλγόριθμοι παραγωγής ονομάτων, Δίκτυα που καθορίζονται από λογισμικό, Προγραμματισμός σε επίπεδο δεδομένων, eXpress Data Path (XDP), Μηχανική Μάθηση, Μηχανισμοί που σέβονται την ιδιωτικότητα

Contents

Acknowledgments/Ευχαριστίες

Η διατριβή εκπονήθηκε στο εργαστήριο Διαχείρισης & Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής (Network Management & Optimal Design Laboratory, NETMODE) του Εθνικού Μετσοβίου Πολυτεχνείου (ΕΜΠ). Το μεγαλύτερο μέρος της ενισχύθηκε σημαντικά από την υποτροφία τετραετούς διαρκείας του Ειδικού Λογαριασμού Κονδυλίων Έρευνας (ΕΛΚΕ) του ΕΜΠ.

Θα ήθελα αρχικά να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου, Βασίλη Μάγκλαρη, για την πολύτιμη καθοδήγηση και την ουσιαστική υποστήριξη σε όλη τη διάρκεια των διδακτορικών μου σπουδών. Ακόμα, τον ευχαριστώ για την ευκαιρία που μου έδωσε να εργαστώ στο απίστευτα ενδιαφέρον πεδίο της ασφάλειας δικτύων υπολογιστών.

Ευχαριστώ πολύ τον Δημήτρη Καλογερά για τις ενδιαφέρουσες συζητήσεις που κάναμε σε θέματα δικτύων υπολογιστών, τις πολύτιμες ιδέες και κατευθύνσεις του για τη διατριβή, καθώς και για την αφιέρωση σημαντικού χρόνου στη βελτίωση των εργασιών μου.

Ευχαριστώ πολύ τη Μαίρη Γραμματικού για τη συνεργασία στα πλαίσια εργαστηρίων της Σχολής μας και την υποστήριξη σε θέματα που αφορούσαν τη διατριβή μου στα πλαίσια σχετικών ευρωπαϊκών ερευνητικών προγραμμάτων. Ευχαριστώ ακόμα τον συνάδελφο, Δημήτρη Πανταζάτο, για την υποστήριξη που προσέφερε, καθώς και για τις πολύ ευχάριστες και ενδιαφέρουσες συζητήσεις τεχνικού και προσωπικού επιπέδου. Επίσης, ευχαριστώ τους Αδάμ Παυλίδη και Μαρίνο Δημολιάνη για τη συνεργασία τους όπου χρειάστηκε. Τέλος, ευχαριστώ τα υπόλοιπα μέλη του NETMODE για ωραίες στιγμές.

Ευχαριστώ πολύ την Αναστασία Δημακοπούλου και τους Σταύρο Κορέντη, Μάριο Μιχαηλίδη, Νίκο Μπαζώτη, Ηλία Μπίμπα και Γιώργο Σουλιώτη με τους οποίους συνεργάστηκα στα πλαίσια των διπλωματικών τους και από τις οποίες προέκυψαν εξαιρετικά αποτελέσματα.

Ευχαριστώ πολύ τους αγαπητούς φίλους Σωτήρη Στάμου και Κωνσταντίνο Ράγιο για το συνεχές ενδιαφέρον τους σχετικά με την πορεία του διδακτορικού μου, καθώς και για ενθαρρυντικά λόγια τους.

Ευχαριστώ πάρα πολύ τους γονείς μου, Γιώτα και Γιώργο, για τη συνεχή φροντίδα τους και τις συμβουλές τους σε διάφορους τομείς της ζωής μου. Τους ευχαριστώ πολύ για την ενθάρρυνσή τους να ξεκινήσω τις σπουδές μου, το ενδιαφέρον που έδειχναν όταν τους μιλούσα για θέματα της δουλειάς μου, καθώς και για την υποστήριξή τους σε δύσκολες στιγμές.

Τέλος, ευγνωμονώ τη σύντροφό μου Παναγιώτα για τη φροντίδα της και την τεράστια υποστήριξη σε όλη τη διάρκεια των διδακτορικών μου σπουδών. Την ευχαριστώ για το ενδιαφέρον που έδειχνε πάντα για τη δουλειά μου, την ουσιαστική ενθάρρυνσή της ότι θα καταφέρω να ολοκληρώσω τις σπουδές μου, την υπομονή της να ακούει πολλά προβλήματα της δουλειάς μου, αλλά και για την ανοχή της σε πολλές περιπτώσεις.

List of Abbreviations

AD Attack Detector

AdaBoost Adaptive Boosting

ADC Anomaly Detection Component

API Application Programming Interface

ASCII American Standard Code for Information Interchange

AXFR Authoritative Transfer

BCC BPF Compiler Collection

BF Bloom Filter

BGP Border Gateway Protocol

BiLSTM Bidirectional Long Short-Term Memory

BMv2 Behavioral Model version 2

C&C Command & Control

CapEx Capital Expenditure

CDN Content Delivery Network

CF Cuckoo Filter

CMS Count-Min Sketch

CNN Convolutional Neural Network

COTS Commercial Off-The Shelf

CPU Central Processing Unit

DC Data Collector

DDoS Distributed Denial of Service

DFW DNS Firewall

DGA Domain Generation Algorithm

DHCP Dynamic Host Configuration Protocol

DLT Distributed Ledger Technology

DNS Domain Name System

DNSSEC Domain Name System Security Extensions

DoS Denial of Service

DPDK Data Plane Development Kit

DPI Deep Packet Inspection

DT Decision Tree

eBPF extended Berkeley Packet Filter

ExtraTrees Extremely Randomized Trees

FlowSpec Flow Specification

FN False Negative

FP False Positive

FPGA Field Programmable Gate Array

FQDN Fully Qualified Domain Name

GB Gradient Boosting

GDPR General Data Protection Regulation

GPU Graphics Processing Unit

GRE Generic Routing Encapsulation

HIDS Host-based Intrusion Detection System

HsZn Hashed DNS Zone

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IDN Internationalized Domain Name

IDS Intrusion Detection System

IETF Internet Engineering Task Force

IncZn Incremental DNS Zone

IoT Internet of Things

IP Internet Protocol

IPC IP Classifier

IPS Intrusion Prevention System

IXFR Incremental Zone Transfer

IXP Internet Exchange Point

LIME Local Interpretable Model-Agnostic Explanation

LSTM Long Short-Term Memory

MAC Medium Access Control

ML Machine Learning

MLE Maximum Likelihood Estimation

MLP Multi-Layer Perceptron

MPLS Multi-Protocol Label Switching

MTC Mitigation Trigger Component

NETMODE NETwork Management & Optimal DEsign

NFV Network Functions Virtualization

NFVI NFV Infrastructure

NIC Network Interface Card

NIDS Network-based Intrusion Detection System

NLP Natural Language Processing

NREN National Research and Education Network

OF OpenFlow

OpEx Operational Expenditure

OS Operating System

OSPF Open Shortest Path First

OvS Open vSwitch

P2P Peer-to-Peer

P4 Programming Protocol-Independent Packet Processors

PAZM Privacy-Aware Zone Manager

PC Personal Computer

PCC Pearson's Correlation Coefficient

PDP Programmable Data Plane

PISA Protocol-Independent Switch Architecture

PltZn Plaintext DNS Zone

PMD Poll Mode Driver

RBT Red-Black Tree

Recursor Recursive DNS Server

RF Random Forest

RR Resource Record

RRD Round-Robin Database

RRDfile Round-Robin Database file

RRDtool Round-Robin Database tool

SDN Software-Defined Network

sFlow sampled Flow

SHAP SHapley Additive exPlanation

SMOTE Synthetic Minority Oversampling Technique

SVM Support Vector Machine

SymSpell Symmetric delete Spelling correction

TCAM Ternary Content-Addressable Memory

TCP Transmission Control Protocol

TLD Top-Level Domain

TLS Transport Layer Security

TN True Negative

ToS Type of Service

TP True Positive

TTL Time To Live

UDP User Datagram Protocol

VLAN Virtual Local Area Network

VM Virtual Machine

XAI eXplainable Artificial Intelligence

XBI eXplainability Background Instance

XDP eXpress Data Path

XDP DPP XDP Data Plane Program

XGBoost eXtreme Gradient Boosting

XTI eXplainability Test Instance

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation & Problem Statement

Domain Name System (DNS) [1] provides mechanisms for mapping symbolic identifiers (i.e. domain names) to various types of information, mainly numeric values corresponding to Internet Protocol (IP) addresses. DNS offers multiple benefits to network administrators. Indicatively, domain names may be used to locate computational resources based on strings, which are usually easy to memorize, or employed for uniquely identifying Internet nodes whose IP addresses frequently change. Most online services depend on domain names for their operation, therefore DNS constitutes an essential component of the Internet infrastructure.

The major importance of domain names for Internet services results in DNS being frequently targeted by cyber-attacks and specifically Distributed Denial of Service (DDoS) attacks. DDoS attacks forward great amounts of malicious traffic to victim servers and/or networks. Their purpose is to deplete victim resources, e.g. available processors, memory and/or bandwidth. Recent reports [2] by leading cybersecurity companies have demonstrated a significant increase in DDoS attacks targeting DNS infrastructures or abusing their vulnerabilities to disrupt critical, non-DNS systems. The impact of such attacks may be severe; victims usually suffer great economic loss, whereas their public image is often harmed.

Several DDoS attack vectors pertaining to DNS have been reported in recent cybersecurity incidents. Amongst them, Water Torture has been one of the most important with devastating consequences on victims. Research interest in Water Torture attacks has widely increased since 2016 when they were among the attack vectors targeting Dyn, a well-known provider of DNS services. Dyn DNS servers were flooded by approximately 100,000 Internet of Things (IoT) devices with malicious traffic reaching a peak at 1.2 Tbps [3]. As a result, online services of major companies, including Amazon, GitHub, Netflix, PayPal, Reddit, Twitter (X since July 2023) and Spotify, became temporarily inaccessible to legitimate users resulting in severe financial impact.

Water Torture attacks [4, 5] target Authoritative DNS Servers, which are name servers that maintain the DNS Resource Records (RR's) of specific zones, i.e. subsets of the DNS infrastructure managed by particular organizations. Water Torture floods Authoritative DNS Servers

with huge volumes of DNS requests to exhaust their computational resources. The main attack characteristic is that these DNS requests involve domain names, which are typically not included within the zones of victim Authoritative DNS Servers. This is accomplished by randomly generating the prefixes (i.e. first labels) of requested names so that they are never repeated. This method enables the attackers to bypass the DNS caches of intermediary Recursive DNS Servers (Recursors); Recursors search for the Authoritative DNS Servers corresponding to the zones that may involve the requested domain names. Thus, all attack requests are forwarded to the victim name server, eventually maximizing Water Torture impact.

Apart from DDoS attacks, DNS may also be associated with other malicious activities that involve significantly less traffic and thus, their detection is harder. As previously mentioned, DNS is an ubiquitous protocol facilitating the operation of most Internet services. Therefore, network administrators usually enforce firewalling policies that do not block DNS traffic so that DNS messages are normally forwarded to their destination. Such policies are frequently exploited by botnet orchestrators who employ DNS to establish communications between compromised hosts (bots) and Command & Control (C&C) servers. Bots leverage on Domain Generation Algorithms (DGA's) [6] to produce several domain names based on a seeding technique known to C&C servers. A small percentage of these names is registered to point towards C&C server IP addresses, whereas the remaining names are not associated with any DNS data. Bots query the generated domain names until they resolve those that will enable them to locate their C&C servers. The typically high number of requested domain names combined with frequent changes in the DGA seeds significantly complicate malicious DNS request detection efforts.

Safeguarding DNS is of paramount importance to network administrators. Traditional security systems for defending against DNS threats, including Water Torture attacks and DGA's, mostly rely on whitelists of legitimate domain names and/or Machine Learning (ML) algorithms. Name whitelists are inventories of valid DNS names; requests pertaining to the whitelisted names are forwarded to their destination, while the rest are dropped as malignant. ML algorithms are tuned based on previously acquired knowledge to differentiate between benign and malicious domain names. Despite their promising results, the aforementioned protection mechanisms involve important limitations and pose significant challenges:

- **Computational resource constraints:** Traditional whitelist-based approaches for filtering DNS traffic mainly employ data structures and algorithms that require the exact, plaintext form of domain names to operate. Name whitelists mapping entire DNS zones may involve millions of entries [7], hence inefficient data structures and algorithms may result in time-consuming element lookups, whereas the total memory consumption may be considerably high. As a result, filtering appliances may be incapable of holding whitelists in-memory, while mitigation throughput may be incompetent for modern high-speed DDoS attacks.
- **Privacy concerns:** Access to Authoritative DNS Server zone contents is generally restricted by network administrators for security reasons [8]. This obstructs the construction of inclusive domain name whitelists resulting in less accurate filtering of DNS messages. Furthermore, DDoS mitigation cannot be enforced closer to the attack origins (e.g. within

Recursors or upstream scrubbing infrastructures), where packet filtering is generally more efficient.

- **Packet filtering performance limitations:** DNS attack mitigation mechanisms mainly rely on Deep Packet Inspection (DPI), i.e. examination of the DNS message payload, for extracting requested domain names. These names are subsequently fed to whitelists or ML algorithms, which classify them as benign or malicious. DPI and name classifications involve costly operations that are traditionally implemented within the user space of filtering appliances. The mitigation throughput of such user-space solutions may be unable to cope with the constantly increasing rate of modern DDoS attacks.
- **Limited understanding of ML model decisions:** ML has been widely investigated as a promising approach to effectively differentiate between legitimate and malicious network traffic. Constant efforts to improve ML model performance resulted in the replacement of simple and intrinsically explainable white-box models by complex, black-box ones. Such black-box ML algorithms are not interpretable, hence their adoption in production environments is hindered; users of network services are unable to receive justifications about model decisions, whereas developers are incapable of verifying the functionality of their models. Thus, network administrators are usually reluctant to incorporate ML algorithms within their attack detection and mitigation systems, eventually relying on traditional protection methods, which may be less effective.

Approaches for overcoming the aforementioned limitations and efficiently protecting against DNS threats may be explored within the fields of Big Data and Software-Defined Networks (SDN's). Specifically, probabilistic data structures and Programmable Data Planes (PDP's) may be promising methods for accelerating cyber-attack detection and mitigation. Furthermore, eXplainable Artificial Intelligence (XAI) algorithms have been proposed to derive interpretations of complex ML models.

Probabilistic data structures [9] are widely employed for Big Data analytics because they perform traditional operations in a time and space efficient manner. These operations may pertain to membership lookups, frequency estimations, median approximation and set similarity. Probabilistic data structures effectively perform such operations by mapping elements hashed instead of their plaintext form; configurable errors are introduced, but the total calculations on stored data are considerably decreased. Probabilistic data structures may be beneficial for DNS-related DDoS attack detection and mitigation; small percentages of erroneous decisions may be tolerated for efficient whitelist lookups and domain name frequency estimations. Furthermore, storing elements hashed makes probabilistic data structures suitable for applications requiring privacy-aware exchanges of sensitive information among participating collaborators.

SDN's have revolutionized computer networking by decoupling the data and control planes of network devices; routing decisions are made within the control plane, whereas data plane is the part of the network where packets are actually forwarded based on the aforementioned decisions. Programmable Data Planes (PDP's) have been recently introduced as a promising

implementation of the SDN paradigm. They enable network administrators to define the forwarding logic and operations of white-box switches by directly programming their chips; this was impossible with the fixed-function chips of legacy networking equipment. Therefore, network administrators are capable of supporting custom protocols and experimental features for DDoS detection and mitigation. Contrary to alternative SDN enablers (e.g. OpenFlow - OF) that mainly rely on external controllers for forwarding decisions and application of security policies, PDP's process packets based on decisions made predominantly within the data plane of networking devices. Thus, frequent communication with external mitigation systems is significantly decreased and line-rate DDoS attack protection may be accomplished. Apart from networking equipment, PDP's also apply to Commercial Off-The Shelf (COTS) hardware (e.g. low-cost Network Interface Cards - NIC's) by providing technologies, which accomplish high-speed packet processing.

Among various PDP implementations, the eXpress Data Path (XDP) framework [10] has recently attracted significant interest. XDP establishes a programmable data path within the Linux kernel and processes ingress packets at an early system level, e.g. at the NIC drivers, based on extended Berkeley Packet Filter (eBPF) programs. The total system operations (e.g. memory lookups) performed on received traffic are thus minimized and the packet processing throughput is significantly increased. Contrary to other data plane programmability technologies, XDP does not bypass the Linux kernel (like Data Plane Development Kit - DPDK [11]) and does not depend on specialized hardware (like Programming Protocol-independent Packet Processors - P4 [12]). Therefore, XDP enables network administrators to utilize functionalities available within the Linux kernel (e.g. TCP/IP code), whereas eBPF programs may run on programmable COTS hardware, e.g. low-cost SmartNIC's.

Although PDP's have revolutionized DDoS detection and mitigation by enabling packet processing at remarkable throughputs, their application in DNS cyber-attacks is not straightforward. PDP's, including XDP, accomplish line-rate packet processing by imposing important constraints on developed programs; unbounded loops are not available, decimal numbers are not supported, whereas the total program size is restricted [10]. Such constraints significantly complicate the data plane implementation of traditional DNS security mechanisms. These are mainly based on DPI for parsing DNS payload, hashing strings and infering ML algorithm outputs, which heavily depend on loops and decimal numbers. Therefore, significant effort is required to apply PDP's for DNS threat detection and mitigation.

Requirements for ML model interpretations led to the development of XAI techniques [13, 14]. Among various XAI methods, SHapley Additive exPlanation (SHAP) [15] is one of the most prominent because of its properties. Specifically, SHAP is a model-agnostic and post-hoc method; it is applicable to all ML models, including complex tree classifiers and deep neural networks, after their learning phase has been completed. Originating from the field of game theory, SHAP determines feature importance by considering how much classification decisions vary when specific feature subsets are utilized. SHAP is capable of delivering both global and local interpretations; global interpretations report the classification criteria of the investigated

ML model on various dataset sample points, whereas local ones detail decision criteria on single dataset instances. Moreover, SHAP involves various visualization tools, which facilitate comprehension of model interpretations. XAI techniques like SHAP are promising for expediting the adoption of ML algorithms in network security applications because they enable network administrators to develop efficient protection services, while justifications on the operation of their deployed models are possible.

1.2 Dissertation Contributions

Driven by the aforementioned challenges, this dissertation proposes mechanisms for effective protection against cybersecurity threats targeting DNS (i.e. Water Torture DDoS attacks) or abusing it to forward malicious data (i.e. Domain Generation Algorithms - DGA's). We overcome the limitations of traditional DNS security systems by leveraging on promising technologies from the fields of Big Data (probabilistic data structures, Machine Learning - ML, eXplainable Artificial Intelligence - XAI) and Software-Defined Networking (PDP's and specifically XDP).

Our key contributions are:

- **DNS Water Torture attack detection and mitigation based on efficient data structures and algorithms:** We leverage on Big Data techniques, widely employed in streaming applications, to effectively detect and mitigate DDoS attacks against the DNS infrastructure, specifically Water Torture attacks. Our proposed systems rely on probabilistic data structures (i.e. Bloom Filters - BF's [16] and Count-Min Sketches - CMS's [17]) to perform name lookups and frequency estimation in a time and space efficient manner. We also utilize deterministic Natural Language Processing (NLP) algorithms (i.e. SymSpell [18]) to rapidly detect spelling mistakes, thus enabling fine-grained classification of requester IP addresses as legitimate and suspicious. Contrary to related approaches, which are constrained by the total number of entries stored within domain name whitelists, our proposed mechanisms are capable of mapping large DNS zones without performance degradation at the cost of a configurable and tolerable error percentage.
- **Acceleration of DPI tasks for efficient DNS DDoS attack mitigation based on data plane programmability methods:** We use PDP's to effectively differentiate between legitimate and malicious DNS messages within the data plane of filtering appliances. Specifically, we use XDP to accelerate DPI tasks traditionally performed within the user space of DNS security systems; domain name parsing, lookups in whitelists based on probabilistic data structures and ML algorithm inference. We develop data plane implementations of the aforementioned tasks tailored to the challenges introduced by eBPF programs, i.e. the lack of decimal numbers, the absence of unbounded loops and the limited program size. Our data plane solutions significantly increase mitigation throughput compared to respective user-space implementations without requiring specialized hardware like other data plane programmability technologies, e.g. P4 switches.

- **Privacy-aware zone exchanges based on probabilistic data structures:** We employ the privacy-aware properties of probabilistic data structures and specifically Cuckoo Filters (CF's) [19] to facilitate zone exchanges between Authoritative DNS Servers and third-party filtering appliances, e.g. Recursors or upstream cloud scrubbing infrastructures. Therefore, mitigation of DNS DDoS attacks (e.g. Water Torture) may be enforced closer to the attack sources, where it is generally more efficient. Moreover, network administrators may rely on third-party whitelists of valid names to filter DNS traffic related to botnet activities (e.g. DGA's) at the edge of their networks with increased accuracy. Contrary to related approaches relying on BF's, which do not support element deletions, our proposed CF-based mechanism enables incrementally updating the maintained whitelists, thus downloading again entire zones is not required.
- **Global and local interpretations of tree and deep neural network DGA name classifiers based on model-agnostic, post-hoc XAI methods:** We leverage on SHAP to derive interpretations of DGA name classifiers in a model-agnostic and post-hoc manner, i.e. independently of the ML model and after the learning process is completed. Based on various SHAP visualization tools (summary, dependence and force plots), we provide global and local interpretations that detail how our DGA name classifiers distinguish legitimate from malicious domain names. Contrary to related approaches, we assess feature importance and feature interactions with respect to well-known fundamental DGA schemes (arithmetic, wordlist, hash and permutation based). Learning and interpretations rely on domain-specific features, which are extracted directly from domain names and report their statistical and linguistic properties. Therefore, time-consuming accesses to external databases including historical information are avoided; extraction of features that rely on historical data typically requires costly operations, while privacy concerns may be raised.
- **Extensive experimental validation of proposed mechanisms:** Our developed security solutions are validated via proof of concept setups deployed within the NETwork Management & Optimal DEsign (NETMODE) laboratory virtualized infrastructure. Experimentation relies on up-to-date datasets widely-used for DNS research (e.g. DGArchive repository [20], Tranco list [21]) and synthetic traffic, generated based on statistics from publicly available data (Booters DDoS-as-a-Service attacks [22]) and experience from our university campus network services.

1.3 Dissertation Outline

The remainder of this dissertation is structured as follows:

- **Chapter 2** provides brief background on important concepts related to this dissertation. We initially elaborate on DNS fundamentals, DDoS attacks and DNS security threats. We subsequently describe SDN's and infrastructure monitoring tools. We then delve into data

structures and algorithms, which will be used to develop efficient DNS protection systems. Finally, we discuss ML fundamentals, basic ML algorithms and XAI techniques.

- **Chapter 3** describes our user-space schema for effective DNS Water Torture attack detection and mitigation. Our solution relies on efficient data structures and algorithms to safeguard Authoritative DNS Servers from malicious DNS traffic. We mainly emphasize on BF's, which are utilized to map entire DNS zones in a space-efficient manner. BF's enable rapid name lookups and effectively distinguish legitimate from malignant DNS requests.
- **Chapter 4** extends our user-space, BF-based protection mechanism for Water Torture attack mitigation within the data plane of Authoritative DNS Servers. We leverage on the XDP framework to efficiently discern legitimate and malicious DNS requests at the NIC driver level of victim Authoritative DNS Servers. Therefore, mitigation throughput is significantly improved compared to our user-space solution described in Chapter 3.
- **Chapter 5** proposes a schema for privacy-aware DNS zone exchanges. We employ CF's to map entire Authoritative DNS Server zones and make them available to third-party filtering appliances (e.g. Recursors or upstream scrubbing facilities). Thus, DNS attack mitigation may be accomplished more efficiently closer to the DDoS attack origins and/or DNS administrators may filter DGA-related traffic more accurately at the edge of their networks.
- **Chapter 6** extends our previous work on data plane DNS attack mitigation. Specifically, we use XDP to accelerate ML classification within the data plane of Recursors. We develop binary Naive Bayes Classifiers tailored to eBPF program constraints. Our developed models effectively differentiate between legitimate and malicious DNS requests at the Recursor NIC driver level.
- **Chapter 7** describes our schema for analyzing the operation of DNS classifiers based on XAI techniques. Specifically, SHAP is employed to derive global and local interpretations of DGA name classifiers in a post-hoc and model-agnostic manner. Therefore, we determine the impact of the utilized features and indicate how their individual values contribute to the classification of DNS names as legitimate or malicious.
- **Chapter 8** summarizes our work and discusses future research directions.
- **Chapter 9** lists our research publications in scientific journals and international conferences/workshops.
- **Chapter 10** summarizes the dissertation in the Greek language.

Chapter 2

Background

This chapter includes theoretical background on important concepts of the dissertation. The chapter is organized as follows:

- Section 2.1 presents Domain Name System (DNS) fundamentals.
- Section 2.2 discusses Distributed Denial of Service (DDoS) attacks.
- Section 2.3 delves into cyber-attacks targeting DNS infrastructures or abusing DNS vulnerabilities to support malicious activities, such as attacks to critical, non-DNS services or communication between infected devices (bots) and their control servers.
- Section 2.4 introduces Software-Defined Networks (SDN's) and explores Programmable Data Planes (PDP's).
- Section 2.5 discusses network and system monitoring methods widely employed in production environments.
- Section 2.6 delves into data structures and algorithms, which will be employed in this dissertation to enable time and memory efficient data operations for DNS cyber-attack detection and mitigation.
- Section 2.7 summarizes Machine Learning (ML) fundamentals, outlines widely-used ML algorithms and discusses eXplainable Artificial Intelligence (XAI) techniques.

2.1 Domain Name System (DNS) - Fundamentals & Operation

DNS [1] maps **domain names** (i.e. symbolic string identifiers) to various types of information pertaining to computational resources, mainly Internet Protocol (IP) addresses, or vice versa. DNS operation is similar to telephone directories that associate subscribers with their phone numbers. Internet users are thus not required to memorize difficult details, such as long numeric addresses; they may rely on resource names and leverage on DNS to obtain the necessary information.

DNS is vital for computer networks. Erroneous DNS configurations or disruptions to its normal operation are usually responsible for rendering major Internet services inaccessible to their

users [23]. Such outages typically cause severe financial damage to impacted organizations, whereas their reputation may be negatively affected [24].

Requirements for redundancy and scalability resulted in constructing DNS as a **distributed** and **hierarchical** naming database. DNS data are replicated across multiple name servers, which may be in different geographical locations, whereas there are no single name servers (i.e. DNS servers) responsible for all the available DNS mappings. This design facilitates DNS management and enables scalable extensions to the existing DNS infrastructure. Moreover, there are no single points of failure and DNS messages may be load balanced across several name servers, thus name resolution latency is significantly reduced.

The term DNS may refer to: (i) the distributed, hierarchical naming database described above and (ii) the application-layer protocol utilized by name servers and Internet users (DNS clients) for resolving domain names to their respective DNS values [23].

In the following, we provide more details pertaining to the DNS distributed and hierarchical database (subsection 2.1.1), describe the name resolution process (subsection 2.1.2), discuss DNS caching (subsection 2.1.3), analyze the Resource Records (RR's) that are most frequently used in DNS (subsection 2.1.4), describe zone transfers (subsection 2.1.5) and elaborate on details related to the DNS application-layer protocol (subsection 2.1.6).

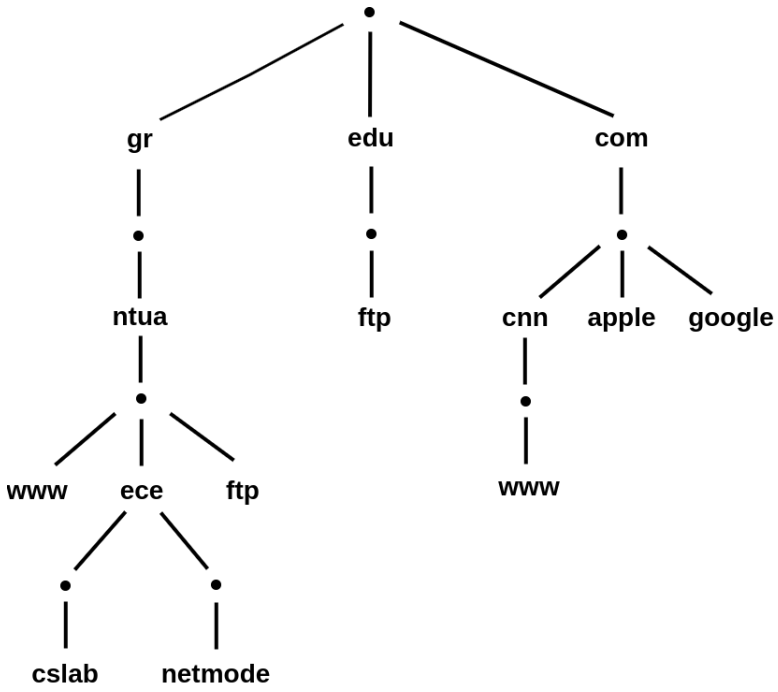

Figure 2.1: DNS is designed as a distributed and hierarchical database - A portion of DNS is depicted

2.1.1 The DNS Distributed & Hierarchical Database

DNS is a distributed and hierarchical naming system consisting of multiple name servers and hosts (e.g. Personal Computers - PC's) organized in a tree data structure. A portion of DNS is depicted in Fig. 2.1. Leaf nodes of Fig. 2.1 correspond to hosts, whereas the remaining ones represent name servers.

Each DNS node (host or name server) is identified by a **Fully Qualified Domain Name (FQDN)**. FQDN's are determined by concatenating all the available labels (e.g. "www" and "edu" in Fig. 2.1) from the considered node up to the DNS root, i.e. the root node of the tree structure. FQDN labels are separated using dot delimiters, whereas a trailing dot may be used to denote the DNS root. Trailing dots are usually omitted because DNS client software appends them automatically to FQDN's when requests are issued to resolve domain names. However, manual addition of trailing dots in configuration files may be required by specific DNS server implementations, e.g. BIND [25].

Indicative FQDN examples constructed based on Fig. 2.1 information are "www.cnn.com." and "netmode.ece.ntua.gr.". The former is built by joining labels "www", "cnn" and "com" moving from the tree child nodes up to the root, whereas the latter consists of labels "netmode", "ece", "ntua" and "gr". Both FQDN's include trailing dots (".") corresponding to the DNS root, which could have been omitted. Note that in the remainder of the dissertation, terms "FQDN", "name" and "domain name" will be used interchangeably.

DNS is segmented into multiple **domains**. Each domain includes a subset of DNS nodes and may be further segmented into subdomains. Mappings between FQDN's and their corresponding values (e.g. IP addresses) are stored as **Resource Records (RR's)** within DNS **zones**. Zones are DNS subsets managed by a specific entity and facilitate the granular management of DNS. Fig. 2.2 depicts a portion of DNS divided into 5 zones; zones are visualized using elliptical shapes.

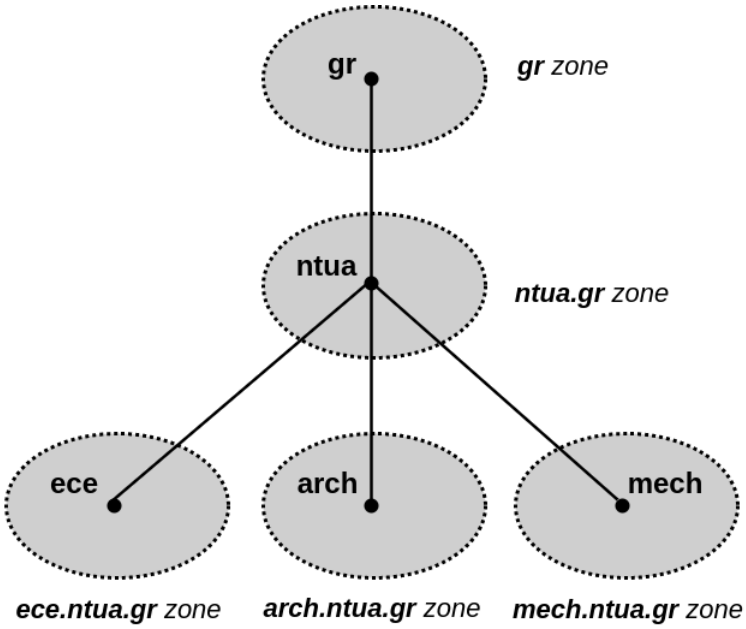

Figure 2.2: Zones are portions of the DNS managed by a particular organization - 5 zones are indicatively depicted

Name servers responsible for maintaining the DNS mappings (i.e. RR's) of specific zones are called **Authoritative DNS Servers**. RR's are contained in **zone files**, whose format depends on the utilized DNS software. Indicatively, BIND [25] stores zone files as text files, whereas RR's in PowerDNS [26] are held in databases. A zone may be served by a single Authoritative DNS Server or multiple for redundancy and load balancing purposes. In case of multiple

instances, zones are updated at the **master/primary** Authoritative DNS Server. Modifications are subsequently propagated to **slave/secondary** Authoritative DNS Servers via zone transfers using specific types of DNS messages (see subsection 2.1.5).

According to their level within DNS [23], Authoritative DNS Servers may be further categorized as:

- **Root DNS Servers:** They are authoritative for the root DNS zone, which corresponds to FQDN trailing dots (see Fig. 2.1). They resolve requests related to the Authoritative DNS Servers of the first DNS level; this includes Top-Level Domain (TLD) zones, e.g. "com" and "edu" zones depicted in Fig. 2.1. There are 13 Root DNS Servers denoted with letters A-M. However, for load balancing and redundancy purposes, each Root DNS Server actually consists of multiple instances, which are distributed across the globe and share an IP address. Requests from DNS clients are distributed to Root DNS Server instances based on Anycast [27], which routes DNS traffic based on the load of name servers and their proximity to DNS clients.
- **TLD Servers:** They are authoritative for the zones of the first DNS level, therefore they resolve requests leading to the Authoritative DNS Servers of the second DNS level. Labels associated with TLD zones often denote country names (e.g. "gr", "fr" and "es") or the type of organizations, e.g. "com" for commercial and "edu" for educational purposes.
- **Lower-level Servers:** They are responsible for requests pertaining to zones corresponding to lower DNS levels.

2.1.2 Domain Name Resolutions

Domain name resolutions require searching the DNS infrastructure to determine the Authoritative DNS Servers that maintain specific DNS RR's; their associated values are eventually returned to DNS clients. The majority of name resolutions in production environments pertain to domain names that correspond to IP addresses.

Although DNS clients are capable of resolving names themselves, this task is usually delegated to **Recursive DNS Servers (Recursors)** for increased flexibility. Such servers perform requests towards multiple Authoritative DNS Servers, traversing DNS in an hierarchical manner until they locate those responsible for the queried DNS information.

Recursors may leverage on either the **iterative** or **recursive** method to traverse the DNS infrastructure [23]. The iterative resolution of name "www.example.com" is depicted in Fig. 2.3. The steps required are:

- **Step 1:** The PC issues a DNS request for the IP address corresponding to the domain name "www.example.com". The request is forwarded to the Recursor.
- **Step 2:** The Recursor issues a DNS request towards a Root DNS Server. Notably, Root DNS Server IP addresses are usually stored within a text file, thus they are already available to Recursors.

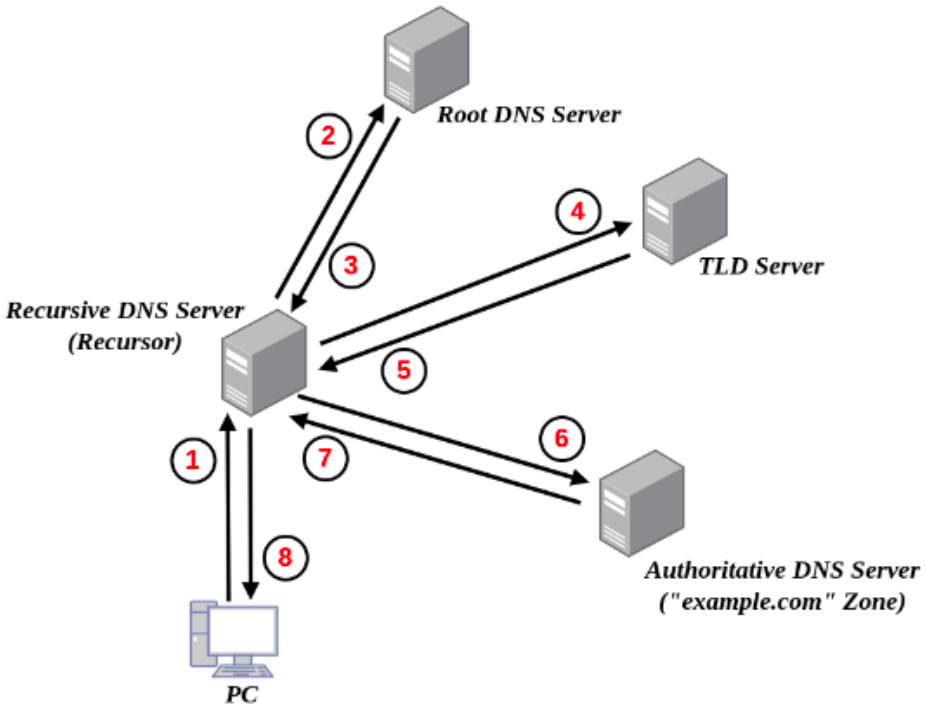

Figure 2.3: Domain name resolutions based on the iterative method - The IP address of FQDN "www.example.com" is determined

- **Step 3:** The Root DNS Server is not authoritative for zones "example.com" and “com”. Therefore, a response is forwarded to the Recursor with the details (i.e. FQDN and IP address) of the TLD servers, which are authoritative for the "com" zone.
- **Step 4:** The Recursor makes a DNS request for the "www.example.com" IP address. The request is forwarded to one of the TLD servers, which are authoritative for the "com" zone.
- **Step 5:** The TLD server is not authoritative for zone "example.com", but for zone "com". Thus, a response is returned to the Recursor with the details (i.e. FQDN and IP address) of the "example.com" zone Authoritative DNS Servers.
- **Step 6:** The Recursor subsequently sends a DNS request for the "www.example.com" IP address to one of the "example.com" zone Authoritative DNS Servers.
- **Step 7:** The "example.com" zone Authoritative DNS Server determines the IP address mapped to the "www.example.com" domain name and responds to the Recursor. If the name is unknown to the Authoritative DNS Server, an error message will be returned.
- **Step 8:** The Recursor responds to the DNS client with the requested information, i.e. the IP address of "www.example.com".

The recursive resolution of "www.example.com" is depicted in Fig. 2.4. In contrast with the iterative method, Recursors will not receive responses from Authoritative DNS Servers unless the RR is found or an error message is returned; Authoritative DNS Servers, which are not responsible for the RR will issue DNS requests to resolve the name themselves. Regardless of the method used (i.e. iterative or recursive), DNS clients perform recursive requests towards

Recursors. However, in production environments, most Recursors rely on the iterative method to issue DNS queries.

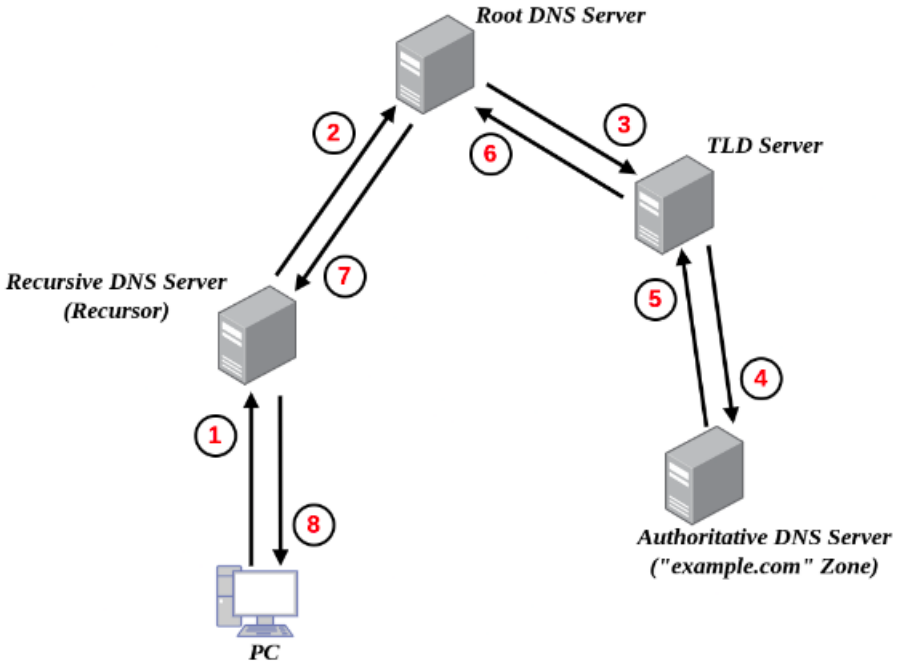

Figure 2.4: Domain name resolutions based on the recursive method - Determining the IP address of the domain name www.example.com

Recursors are typically restricted to the DNS clients of a specific organization for security purposes. However, a limited number of them is configured to respond to the requests of any Internet user; these servers are called **Open Resolvers**. Although such services may be useful for home users or small organizations that wish to avoid local setups, they are often abused for malicious purposes [28] that will be discussed in subsection 2.2.

2.1.3 DNS Caching

Domain name resolutions are time-consuming because they require traversing the DNS infrastructure to locate the appropriate Authoritative DNS Servers. Therefore, upon receiving responses to DNS client requests, Recursors should ensure that name resolutions are not repeated if the same information is needed in a short time period.

Recursors reduce the total number of name resolutions by maintaining a cache memory (DNS cache), which temporarily stores obtained information when a response is received [23]. This information is stored for a time period determined by the Time To Live (TTL) value, which is defined within Authoritative DNS Server zone files. When Recursors receive requests from DNS clients, they first check if the RR is included within their DNS cache. If the information is already available, responses are returned to DNS clients without searching the DNS infrastructure, otherwise Recursors traverse DNS as described in subsection 2.1.2.

TTL value selection [29] is hard and is often based on how frequently the value mapped to a DNS name is expected to change. Incorrectly selected TTL values may cause severe problems.

Indicatively, if the IP address mapped to a domain name changes and a previous value has already been cached by a Recursor, DNS clients will be receiving erroneous responses until the TTL value expires. TTL values corresponding to DNS mappings related to host and server IP addresses are usually set to 86,400 seconds (i.e. 1 day), while smaller values (e.g. 10 minutes) are preferred for names related to web services or Content Delivery Networks (CDN's). IP addresses pertaining to web services or CDN's change frequently, thus lower TTL values are recommended.

2.1.4 DNS Resource Records (RR's)

As mentioned in subsection 2.1.1, Resource Records (RR's) map FQDN's (domain names) to their respective DNS values and vice versa. RR's are stored within Authoritative DNS Server zone files and there are multiple types of them.

The most common DNS RR types [1] are:

- **A:** Maps names to IPv4 addresses, i.e. addresses of 32 bits.
- **AAAA:** Maps names to IPv6 addresses, i.e. addresses of 128 bits.
- **NS:** Lists the Authoritative DNS Servers responsible for a given zone.
- **MX:** Lists the mail servers for a given zone.
- **PTR:** Used to map IP addresses to their associated names, i.e. PTR is the opposite of A and AAAA RR's.
- **SOA:** Provides information about a specific DNS zone, such as the primary Authoritative DNS Server, zone administrator contact details, the default TTL value of the zone RR's and the serial number, which tracks modifications to zone contents.

DNS requests for the aforementioned RR types typically utilize User Datagram Protocol (UDP) at the transport layer. Requests target port 53 of Recursive and/or Authoritative DNS Servers, while source ports are randomly selected.

2.1.5 Zone Transfers

Zone transfers enable DNS clients to retrieve entire DNS zone contents or portions of them. Zone transfers are mostly employed by secondary Authoritative DNS Servers to retrieve zone modifications from their respective primary Authoritative DNS Servers.

DNS requests used for zone transfers leverage on Transmission Control Protocol (TCP) at the transport layer (targeting port 53 of DNS servers) because requested information should be delivered reliably. The DNS requests utilized for zone transfers are:

- **Authoritative Transfer (AXFR) [30]:** Used to retrieve entire zone contents from an Authoritative DNS Server.
- **Incremental Zone Transfer (IXFR) [31]:** Used to retrieve all the zone RR's that have been modified since a previous time moment. Different zone versions are determined by the serial number identifier.

Apart from legitimate DNS clients, zone transfers may also be used by attackers to gain insight into the contents of specific DNS zones; sensitive information pertaining to the users, servers and services of an organization may be exploited by attackers for malicious purposes [8]. Therefore, AXFR and IXFR requests are usually restricted to trusted DNS clients, such as zone secondary Authoritative DNS Servers.

2.1.6 The DNS Protocol

Communication between DNS nodes relies on the DNS protocol. Messages issued by DNS clients to resolve domain names are called DNS requests. The results of these requests are returned to DNS clients within DNS responses.

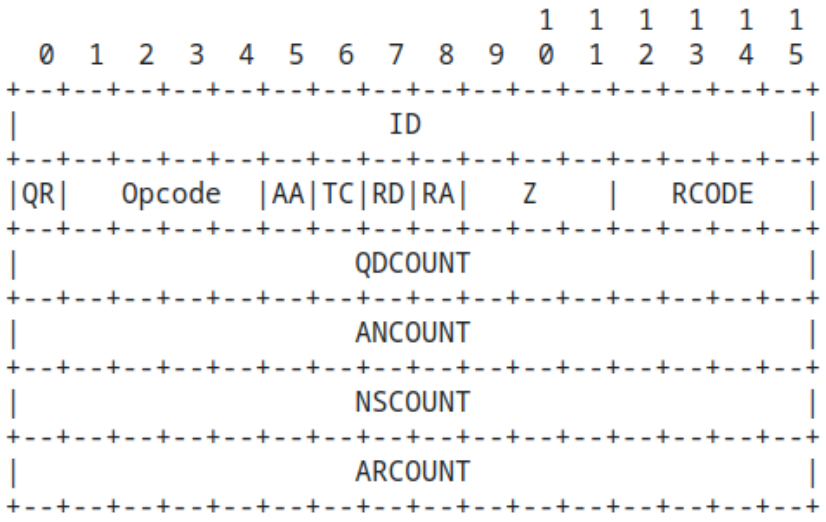

Figure 2.5: Header of DNS requests/responses (figure retrieved from [1])

DNS is an application layer protocol of the TCP/IP Stack. The header of DNS messages is fixed at 12 Bytes, whereas the length of their payload is variable [1]. The DNS message header is depicted in Fig. 2.5 and consists of the following fields:

- **ID - Identification** (2 Bytes): Identifies DNS messages and provides a method to match DNS requests to their corresponding responses.
- **QR - Query/Response** (1 bit): Differentiates between DNS requests (set to 0) and DNS responses (set to 1).
- **OPCODE** (4 bits): Specifies the type of DNS requests, e.g. if the message involves a standard query or whether it is linked to Dynamic DNS updates.
- **AA - Authoritative Answer** (1bit): Set to 1 if the response originates from an Authoritative DNS Server, otherwise AA is set to 0.
- **TC - Truncated** (1 bit): Set to 0 if the entire DNS response is returned. Otherwise, the TC flag is set to 1 if only the initial 512 Bytes of a DNS response are returned.

- **RD - Recursion Desired** (1 bit): Set to 1 from the DNS client to request name resolution based on the recursive method. RD is set to 0 to specify that iterative name resolution is preferred.
- **RA - Recursion Available** (1 bit): Set to 1 by the DNS server responding to a request to denote that recursive name resolutions are supported. Otherwise, the RA flag is set to 0.
- **Z - Zero** (3 bits): Reserved bits, set to 0.
- **RCODE - Return Code** (4 bits): Set to 0 for DNS requests or to a value between 0-15 for DNS responses. Table 2.1 summarizes indicative RCODE values [32].
- **QDCOUNT - Query Count** (2 Bytes): Specifies the number of entries within the message question section, i.e. the number of name resolutions requested by the DNS client.
- **ANCOUNT - Answer Count** (2 Bytes): Determines the RR number within the message answer section, i.e. the number of returned DNS mappings within a response.
- **NSCOUNT - Authority Count** (2 Bytes): Specifies the number of RR's within the message authority section, i.e. the number of Authoritative DNS Servers returned as responsible for the zone pertaining to the requested name.
- **ARCOUNT - Additional Information Count** (2 Bytes): Determines the RR number within the message additional section, which includes additional DNS mappings that were not directly requested by the DNS client. Such RR's may provide useful information, e.g. the IP addresses corresponding to the name servers included in the message authority section.

Decimal Value	RCODE Name	Description
0	NOERROR	There are no errors in the DNS response
1	FORMERR	The DNS request was inappropriately constructed, hence the server was incapable of responding
2	SERVFAIL	The server could not respond because of an operational issue
3	NXDOMAIN	The requested name does not exist within the zones of the respective Authoritative DNS Server
4	NOTIMP	The opcode value included within the received DNS request is not supported by the specific DNS server
5	REFUSED	The DNS server refused to respond to the request

Table 2.1: Indicative RCODE values for DNS responses

2.2 Distributed Denial of Service (DDoS) Attacks

Denial of Service (DoS) attacks aim to prevent legitimate Internet users from accessing online services for short or long time periods. DoS attacks involve a single device, which floods the victim with a great volume of network traffic to exhaust its resources, e.g. processing capacity and/or physical memory (see Fig. 2.6). Overwhelmed by the vast amount of malicious messages, the attack victims are unable to process all the messages originating from legitimate Internet users. Thus, benign users are blocked from crucial network services.

Figure 2.6: Denial of Service (DoS) Attacks - The attacker floods the victim with superfluous Internet traffic to disrupt legitimate Internet users from using online services

DoS attacks do not infiltrate victim systems to execute malicious software (malware) and perform malignant actions, such as credential theft. They benefit from the limited capabilities of attacked infrastructures and exploit vulnerabilities of widely-used network protocols (e.g. DNS and TCP) to deplete victim resources and disrupt the normal operation of online services.

Although DoS attacks were a major threat in the past, a single attacking device is usually incapable of overloading the computational resources of modern hosts and servers. Therefore, DoS attacks have evolved into **Distributed DoS (DDoS) attacks**. In DDoS attacks, a huge number of Internet devices (e.g. 100,000 in Dyn cyberattack [3]) simultaneously forward malicious traffic to one or more victims. Therefore, online services may rapidly become inaccessible to legitimate Internet users (see Fig. 2.7). Apart from the processors and the physical memory of targeted systems, DDoS attacks may also deplete the bandwidth of the network connections leading to the victim.

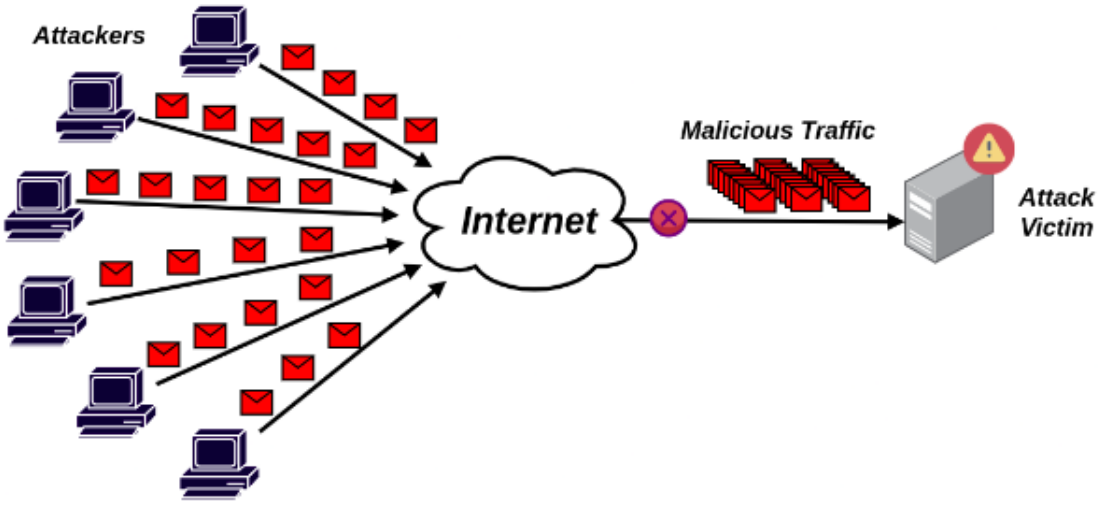

Figure 2.7: Distributed Denial of Service (DDoS) Attacks - The victim is simultaneously attacked by multiple devices, thus networking and computational resources are rapidly depleted

The main DDoS attack consequences [33, 34] involve severe financial loss because most

modern organizations heavily depend on online services to support their clients, whereas the image and trustworthiness of the targeted companies may be harmed. DDoS attacks may be motivated by (i) political, social and/or religious causes of hacktivists, (ii) financial gain often achieved by demanding ransom in order not to attack the victims, (iii) hackers testing or demonstrating their skills and (iv) hacker personal pleasure.

In the following, we discuss important DDoS attack aspects, namely IP spoofing (subsection 2.2.1), botnets (subsection 2.2.2), different DDoS attack types (subsection 2.2.3) as well as commonly employed detection (subsection 2.2.4) and mitigation techniques (subsection 2.2.5).

2.2.1 IP Spoofing

DDoS attack orchestrators usually fake the identity of the attacking devices; their IP addresses are replaced (spoofed) with those of other, potentially legitimate, hosts. This technique is called **IP spoofing** [35] and may benefit the attackers in multiple ways:

- **Identity protection:** Attackers leverage on IP spoofing to conceal their identity, thus avoiding detection by the DDoS attack victims and intermediary protection systems. If the original IP addresses of the attacking devices were used, appropriate mitigation measures (e.g. IP reputation scores) would be promptly deployed to safeguard the attacked systems.
- **Reflection:** Various DDoS attack vectors spoof the source addresses of malicious requests with the victim IP address. Requests are subsequently sent to appropriately selected servers that forward malicious responses to the victim, rendering its offered services inaccessible to legitimate Internet users. This technique is called **reflection**. Reflection is often combined with **amplification**, where small requests result in significantly larger responses. The ratio of the response size to the request size constitutes the attack amplification factor.
- **Response avoidance:** Without IP spoofing, responses to malicious requests will be forwarded to the attacker. Therefore, apart from the victim, the attacker will also suffer the DDoS attack consequences.

2.2.2 Botnets

DDoS attack impact is partly determined by the total number of devices utilized to flood the victim; more devices generally result in increased attack effectiveness. Modern DDoS attacks usually depend on hundred-thousands or even millions of devices to overpower their victims. Attackers gather the necessary capacity by infiltrating devices and installing malware to accordingly gain their control. Such vulnerabilities are usually present in Internet of Things (IoT) devices, which often have poor or no security measures, e.g. appropriately configured firewalls. Notably, apart from DDoS attacks, these hacked devices may also participate in other fraudulent activities, including malware propagation, credential theft and cryptocurrency mining.

Hacked devices are called **bots** (or **zombies**) and the entity controlling them is the **botmaster**, whereas the networks involving bots are called **botnets** [36]. Zombies usually do not commu-

nicate directly with the botmaster, but with intermediary servers (i.e. **Command & Control** or **C&C** servers) distributed across the Internet. C&C servers may serve multiple purposes, such as instructing bots to initiate DDoS attacks, retrieving information collected by infected devices and/or updating the malware installed at bots.

Botnets based on C&C servers adhere to the client-server botnet architecture, which is depicted in Fig. 2.8. Botnets may also operate in a distributed manner without relying on dedicated C&C server instances. Specifically, in **Peer-to-Peer** (**P2P**) botnets [37], bots may simultaneously act as C&C servers forwarding instructions to their peers and clients receiving commands from them. P2P architecture renders botnets more resilient to takedown efforts than traditional, client-server botnets described above.

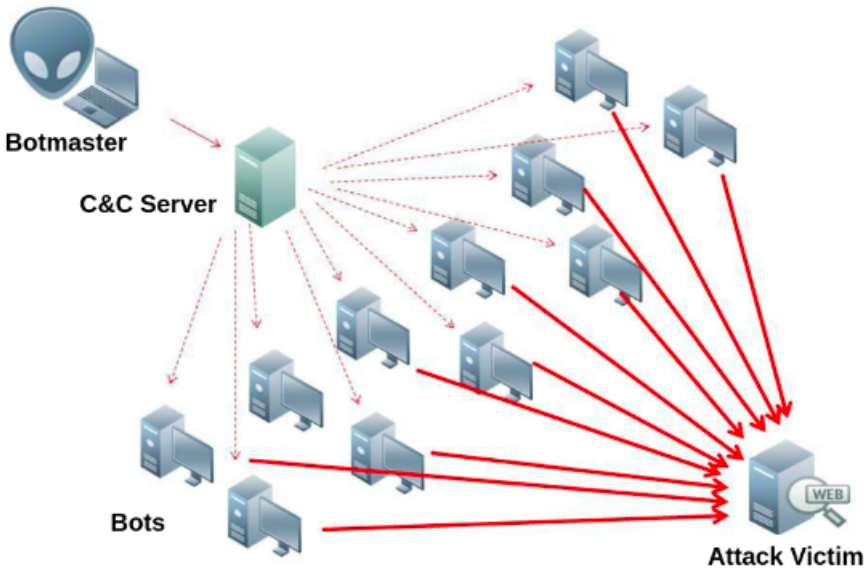

Figure 2.8: Client-server botnet architecture for executing DDoS attacks (figure based on [38])

2.2.3 DDoS Attack Categories

DDoS attacks are broadly categorized in three classes [39] according to the type and volume of attack traffic forwarded to the victim as well as the targeted system vulnerabilities. These classes are:

- **Volumetric attacks:** They flood victims with malicious traffic to congest the bandwidth of the links leading to them. Volumetric attacks typically leverage on reflection and amplification techniques to increase their impact. They are usually reported in bits per second (bps). An indicative volumetric attack is DNS Amplification, which will be described in subsection 2.3.1.3.
- **Protocol attacks:** They target vulnerabilities of network and/or transport layer protocols. Protocol attacks disrupt the services of victims by draining their available resources, e.g. Central Processing Unit (CPU) cores, physical memory, listening sockets, etc. These attacks are usually reported in packets per second (pps). An indicative protocol attack is

SYN Flood [39], which floods victims with TCP SYN segments. Source IP addresses of the messages are spoofed, rendering victims unable to complete TCP handshakes; the available memory of victim servers is entirely consumed by the entries allocated for each incomplete TCP handshake.

- **Application attacks:** They exploit vulnerabilities of application layer protocols to deplete victim resources. These attacks are usually reported in requests per second. An indicative application DDoS attack is Hypertext Transfer Protocol (HTTP) Flood [39], which overwhelms web servers with excessive HTTP requests.

Victims may be simultaneously targeted by various DDoS attack vectors belonging to one or more of the aforementioned categories. These attacks are called **multi-vector attacks** and are generally more effective in depleting victim resources [40] because they are harder to detect and mitigate than attacks based on a single vector.

2.2.4 DDoS Attack Detection Methods

Intrusion Detection Systems (IDS's) are defense solutions used for detecting DDoS attacks or other cybersecurity threats [41]. IDS's capable of mitigating the detected threats are also called Intrusion Prevention Systems (IPS's) [41].

Widely-used IDS categories [41] include Host-based IDS's (HIDS's) and Network-based IDS's (NIDS's). HIDS's are software applications installed within end hosts that monitor system activities to detect malignant behavior. NIDS's are security appliances inspecting network traffic to detect abnormal patterns; they may consist of a single device or multiple devices distributed across the monitored network.

Although IDS's may employ various techniques for DDoS attack detection [41,42], the most widely-used are:

- **Threshold-based:** DDoS attacks are detected when one or more monitored metrics exceed a predefined value (threshold). Monitored metrics may involve single network variables (e.g. the total volume of ingress packets) or the ratio of network variables (e.g. the total number of incoming packets to the total number of outgoing packets). Threshold-based approaches facilitate rapid DDoS attack detection. However, threshold value determination is usually hard [43,44]; multiple false alarms may be raised, whereas low-rate attacks may not be detected if the selected thresholds are not violated [45].
- **Signature-based:** Network traffic is inspected to identify patterns included in a database of malicious signatures. Signature-based approaches enable DDoS attack detection with high accuracy [46]. However, they are unable to detect zero-day threats, i.e. attack vectors that have not been observed before and their signatures are thus unknown [41]. Furthermore, frequent database lookups may prove time-consuming, rendering defense systems unable to cope with modern DDoS attack rates, whereas the total number of stored signatures may be limited by the available IDS memory.

- **Anomaly-based:** Network traffic features observed during a specified time window are compared with those of a baseline traffic profile. DDoS attacks are detected when the inspected traffic characteristics deviate significantly from the baseline model. There are multiple anomaly-based detection methods reported in the literature. The most widely employed include: (i) entropy-based methods (e.g. Shannon and Tsallis entropies [47]), which originate from Information Theory and evaluate the randomness of monitored traffic feature values, (ii) statistical-based methods, which compare the statistical measures of inspected traffic with those of the baseline traffic and (iii) Machine Learning (ML) techniques, which leverage on previous knowledge extracted from past network traffic to generalize to newly observed traffic. Although anomaly-based methods are harder to implement than the aforementioned techniques (threshold-based or statistical-based) and may require longer time intervals to categorize inspected traffic, they are capable of detecting zero-day attack vectors.

2.2.5 DDoS Attack Mitigation Methods

Constraining DDoS attacks after the detection of abnormal traffic is not straightforward. Network administrators should deploy appropriate mechanisms to effectively mitigate attack traffic. The purpose of these mechanisms is to differentiate between legitimate and malicious packets; malicious traffic is dropped, while legitimate traffic is forwarded to its destination.

There are multiple DDoS attack mitigation methods [33, 39, 48, 49]. Some of them are:

- **Rate Limiting:** The ingress and/or egress packet number is monitored over a specified time interval. If the number of packets exceeds a specified threshold, the remaining traffic is dropped. Rate limiting may be applied to specific IP addresses or traffic categories, e.g. DNS requests. Despite being easy to implement and time/memory efficient, rate limiting does not differentiate between legitimate and malicious traffic, therefore large amounts of benign packets may be dropped as a side effect.
- **Blackholing:** After the DDoS attack victim (a single host or an entire subnet) is determined, all traffic destined to the victim is redirected to a null route (blackhole), where it is dropped. Redirection may be accomplished by appropriate Border Gateway Protocol (BGP) route announcements. Although blackholing may effectively protect network services that are not targeted by the DDoS attack, both benign and malicious packets destined to the victim are dropped; the attack purpose is thus accomplished because the victim becomes unavailable to legitimate users. Blackholing is widely used by organizations that do not employ other, more sophisticated, DDoS attack protection systems.
- **BGP Flow Specification (FlowSpec):** Contrary to the aforementioned mitigation solutions, FlowSpec [50] enables fine-grained traffic filtering by matching packets based on their corresponding flows. Flows group packets based on specific header fields; a widely-adopted flow definition involves the source-destination IP, the source-destination ports and the protocol used (TCP, UDP or Internet Control Message Protocol - ICMP). After deter-

mining the characteristics of a detected DDoS attack, the victim may leverage on BGP announcements to propagate mitigation instructions to appropriately configured devices, which will match ingress packets based on specific flows. Packets belonging to these flows may be dropped, rate-limited or redirected to defense systems for further inspection. However, despite the offered flexibility, FlowSpec may be unsuitable for DDoS attacks relying on variable-length fields within the packet payload. An indicative example is DNS Water Torture attack [5], which floods Authoritative DNS Servers with randomly generated domain names of diverse lengths (described in subsection 2.3.1.2).

- **Over-provisioning:** By dedicating more bandwidth and processing resources than necessary to an online service, the victim may be able to sustain high-volume DDoS attacks. However, the constantly increasing rate of modern DDoS attacks may require extensive over-provisioning, which leads to significantly high Capital Expenditures (CapEx) and Operational Expenditures (OpEx).
- **Machine Learning (ML):** Apart from DDoS attack detection, ML algorithms may also be employed for mitigation purposes, i.e. for differentiating between legitimate and malicious network traffic with high accuracy. Although ML inference is usually accelerated via Graphics Processing Units (GPU's), extensive investments in hardware may be required to achieve satisfactory throughput. Nevertheless, the accomplished filtering throughput may be unable to cope with the increasing rate of contemporary DDoS attacks.
- **Deep Packet Inspection (DPI):** Various DDoS attacks (e.g. DNS-based) may require analyzing the packet payload to categorize network traffic as legitimate or malicious. The technique that examines the packet payload is called Deep Packet Inspection (DPI). Although DPI is valuable for mitigating specific DDoS attack types, extensive processing is required per packet, which may be unsuitable for high-speed DDoS attacks. DPI techniques are also useless when traffic encryption is employed.

Small organizations often lack the equipment, human resources and/or expertise required for effectively mitigating DDoS attacks. Such organizations may employ the services of network security companies (e.g. Cloudflare, Imperva, Radware, etc.) or Internet Exchange Points (IXP's), which leverage one or more of the aforementioned mitigation techniques to protect their clients. Upon detecting a DDoS attack, network traffic destined to the victim may be redirected to upstream, cloud-based **scrubbing facilities** [48] of such security companies. The redirected traffic is subsequently filtered to alleviate victim networks from the attack consequences; malicious traffic is dropped, while legitimate packets are forwarded to the clients. Such cloud-based solutions are effective in mitigating attack traffic and thus, they are highly popular among small organizations. However, the cost of subscribing to such security services is usually high.

2.3 DNS Security Threats

DNS is vital for the normal operation of the Internet, enabling users to map names to computing resources. Therefore, DNS is frequently targeted by DDoS attacks aiming to disrupt critical

online services. The connectionless nature of DNS (mostly UDP-based) also renders Recursors a suitable relay for reflection DDoS attacks, whereas firewalling policies that usually do not block DNS messages are exploited by attackers to transmit malignant data within the DNS payload.

In the following, we discuss important aspects of DNS security, specifically DDoS attacks related to DNS (subsection 2.3.1) and Domain Generation Algorithms - DGA's (subsection 2.3.2).

2.3.1 DDoS Attacks Related to DNS

In the following, three important DNS DDoS attack vectors are described, namely DNS Flood (subsection 2.3.1.1), DNS Water Torture (subsection 2.3.1.2) and DNS Amplification (subsection 2.3.1.3).

2.3.1.1 DNS Flood

DNS Flood [51] attacks are UDP Flood variants that target Recursors or Authoritative DNS Servers to exhaust their resources or congest the bandwidth of the links leading to them. Relying on IP spoofing and large botnets, attackers flood the victims with a huge number of valid or invalid DNS requests. By depleting the victim resources, the majority of legitimate DNS clients are incapable of receiving responses to their requests.

2.3.1.2 DNS Water Torture

DNS Water Torture attacks [4, 5, 52] target Authoritative DNS Servers to exhaust their computational resources (e.g. CPU cores). They are also known as DNS slow drip attacks and pseudo-random subdomain attacks.

A Water Torture attack is depicted in Fig. 2.9. A huge number of spoofed DNS requests is generated by the botnet of the attacker. These requests are forwarded to the victim Authoritative DNS Server; requests are relayed via various intermediary Recursors, which may either be local Recursors or Open Resolvers.

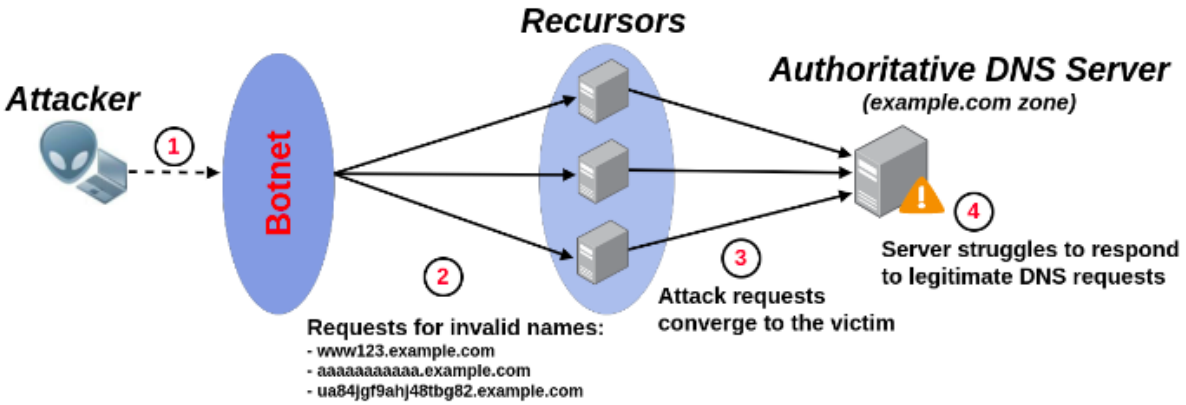

Figure 2.9: The DNS Water Torture Attack

The Water Torture attack main characteristic is that DNS requests are crafted in a randomized

manner. Attackers ensure that the FQDN included in the question section of these requests is invalid, i.e. not included in the victim zones; NXDOMAIN responses are returned by the Authoritative DNS Server. Furthermore, requests for the same domain names are never repeated. This method enables the attackers to bypass the DNS caches of the intermediary Recursors and forward the entire attack traffic to the victim.

Apart from depleting Authoritative DNS Server resources, Water Torture attacks may also degrade intermediary Recursor performance as a collateral damage. As the number of distinct domain names requested by the attacker increases, the number of NXDOMAIN response entries stored within Recursor DNS caches will also raise. Thus, DNS cache lookup time will be higher and the consumed memory resources will significantly increase.

Water Torture attacks can be easily identified via threshold-based detection approaches. The monitored metrics may include the total number of DNS requests received by the victim Authoritative DNS Server, the total number of NXDOMAIN responses returned by the victim and/or the victim CPU utilization. An increased number of NXDOMAIN responses combined with high CPU utilization are clear indications of Water Torture attacks.

Real-time Water Torture mitigation is not straightforward. Network administrators usually mitigate such attacks by rate limiting the total number of responses returned by the victim Authoritative DNS Server. However, rate limiting may result in the rejection of multiple legitimate DNS responses. Machine Learning (ML) algorithms may be used instead for differentiating between valid and invalid names with increased accuracy [5, 53]. However, ML inference is usually implemented at the user space of mitigation appliances, thus the accomplished filtering throughput may be unsuitable for modern, high-speed DDoS attacks.

DDoS attacks, including Water Torture, are typically mitigated more effectively closer to their origins. Traffic filtering may occur in upstream cloud-based scrubbing services or by appropriately deployed filters within Recursors. Accomplishing this requires that whitelists of valid names, constructed from Authoritative DNS Server zone contents are available to upstream mitigation appliances. However, as mentioned in subsection 2.1.5, zone transfers are generally restricted for security reasons and thus, such whitelists are hard to obtain.

2.3.1.3 DNS Amplification

DNS Amplification [54] is a volumetric DDoS attack that depletes the victim network available bandwidth to constitute an online service inaccessible to legitimate Internet users. Notably, attack victims are not limited to DNS services; web servers or other critical services may also be targeted. DNS Amplification is among the most effective attack vectors and the consequences to victims are severe.

A DNS Amplification attack is illustrated in Fig. 2.10. The attacker leverages on a botnet to forward a great number of DNS requests to various Open Resolvers or local Recursors. Using Reflection, the attacker spoofs the source IP address of the requests with that of the victim server. Therefore, DNS responses are forwarded to the victim server instead of the bots that made the respective DNS requests (Reflection). Besides being vital for the attack execution, Reflection

further conceals attacker identity.

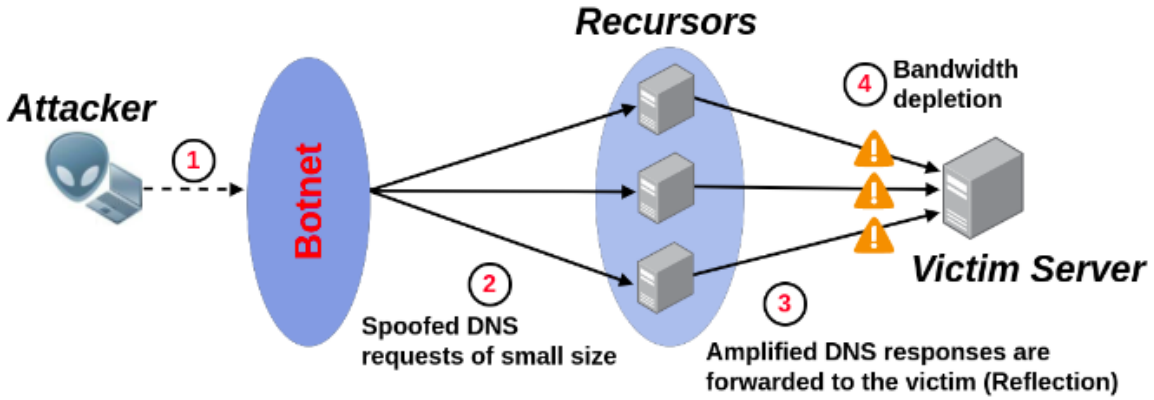

Figure 2.10: The DNS Amplification Attack

The main reason behind the great effectiveness of this attack is Amplification. Attackers benefit from zone RR's with large amplification factors, i.e. names for which requests with small size result in responses with significantly larger size. Such RR's may be of type ANY or related to Domain Name System Security Extensions (DNSSEC) [55, 56], a DNS extension enabling cryptographic authentications for exchanged data. Large amplification factors (e.g. 70 [57]) enable attackers to inflict greater damage to their victims with less resources available, i.e. botnets may comprise of less bots.

Detection of DNS Amplification requires monitoring network traffic for abrupt increases in the number of unsolicited DNS responses. Apart from the techniques outlined in subsection 2.2.5, effective mitigation of DNS Amplification attacks may involve determining responses that do not correspond to previously issued DNS requests.

2.3.2 Domain Generation Algorithms (DGA's)

DGA's are a common technique for establishing communications between hacked devices, i.e. bots, and their orchestrators, i.e. Command & Control (C&C) servers. Bots generate DNS requests based on a seeding technique that is known to C&C servers. A small number of domain names is registered and bots are expected to request their resolution. These names correspond to valid C&C IP addresses, thus bots are capable of locating them. Specifically, bots perform several DNS requests; although most of these requests involve invalid domain names (i.e. NXDOMAIN responses are returned), a limited number of them is successfully resolved to the C&C IP addresses. This typically large number of queried domain names combined with constant changes to the seed render domain name blacklists ineffective.

Therefore, ML algorithms have been suggested as an alternative solution to blacklisting. They leverage on previous knowledge and generalize to newly observed domain names for differentiating between benign and malignant patterns, thus blocking communication between bots and their C&C servers. Notably, various classification algorithms have been investigated with promising results, including deep neural networks [58–63] as well as tree-based classifiers (e.g.

Random Forests - RF's), Support Vector Machine (SVM) and Naive Bayes [64–68].

The seeding strategy, the number of domain names produced by a bot and their structure are determined by the DGA family. Although there are various families with diverse characteristics, DGA's are grouped into the following four generation schemes [6] based on the technique utilized to produce domain names:

- **Arithmetic-based:** These algorithms generate sequences of random values. DGA names are constructed by concatenating the American Standard Code for Information Interchange (ASCII) representations corresponding to these values or using them to locate characters within lists that constitute the DGA alphabet.
- **Wordlist-based:** DGA names are generated by randomly concatenating dictionary words. Thus, domain name randomness is reduced, rendering malicious name detection more complicated.
- **Hash-based:** Domain names are constructed by hashing alphanumeric strings and returning their hexadecimal representation.
- **Permutation-based:** They generate at random a domain name, which is subsequently permuted several times to produce multiple DGA names.

2.4 Software-Defined Networking & Programmable Data Planes (PDP's)

This section delves into the fundamentals of Software-Defined Networks (SDN's) and Programmable Data Planes (PDP's). These technologies have revolutionized network management offering multiple advantages, but they have also introduced new challenges.

In the following, we describe networking planes (subsection 2.4.1), legacy networks (subsection 2.4.2), SDN's and their advantages (subsection 2.4.3), the OpenFlow (OF) protocol (subsection 2.4.4) and Programmable Data Planes - PDP's (subsection 2.4.5).

2.4.1 Networking Planes

Analyzing computer networks in networking planes [69] enables their granular design and administration. Each plane is related to particular aspects of the network and thus, specific types of traffic. Computer networks consist of the following three planes: (i) Data plane, (ii) control plane and (iii) management plane. Their operations are:

- **Data plane:** This plane is responsible for forwarding data between network devices based on the decisions made by the control plane.
- **Control plane:** This plane is responsible for the operation of routing protocols, e.g. Open Shortest Path First (OSPF) and Border Gateway Protocol (BGP). Therefore, the control plane determines how network traffic will be forwarded at the data plane. This plane also involves packet signaling, e.g. Virtual Local Area Network (VLAN) tags, Multi-Protocol Label Switching (MPLS) labels and IP headers, which are necessary for making forwarding decisions.

- **Management plane:** This plane incorporates the network administrator policies that regulate how forwarding decisions are made by the control plane.

2.4.2 Legacy Networks

Legacy computer networks consist of hardware appliances, which are dedicated to specific operations, e.g. routing, switching, security purposes, etc. Device capabilities are specified by their chips; introducing novel features to these devices requires actions from the chip manufacturers, a process that usually requires significant time and effort. Moreover, the data plane of legacy devices is coupled with their control plane. This significantly complicates network administration; applying high-level policies across the network is difficult, automating device configuration is not simple, whereas scaling networks is not straightforward.

2.4.3 Software-Defined Networking Overview

Limitations of legacy networks led to the introduction of Software-Defined Networking [69–71], a novel computer networking paradigm. Fig. 2.11 depicts the major differences between legacy networks and Software-Defined Networks (SDN's). In SDN's, the data plane of network devices is decoupled from their control plane. Routing decisions and/or other advanced control plane operations occur at the **SDN Controller**. Contrary to legacy networks, data plane devices are turned into naive appliances, which mainly forward received frames towards their destination. Their behavior (e.g. where ingress packets are forwarded) is programmed by the control plane, thus they are usually called **programmable switches**.

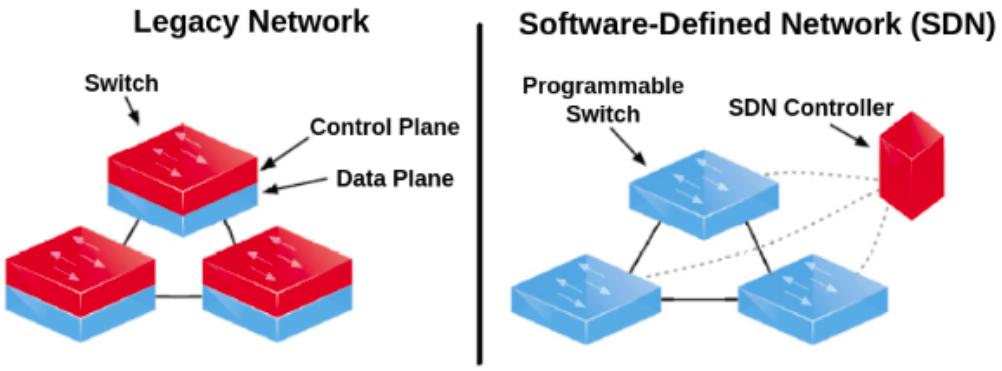

Figure 2.11: Differences between legacy networks and Software-Defined Networks (figure based on [72])

The SDN Controller may be a single, centralized server or may consist of multiple instances distributed across the managed networks. As depicted in Fig. 2.12, the SDN Controller leverages on Application Programming Interfaces (API's) to communicate with the diverse SDN components. Specifically, communication between the SDN Controller and the SDN applications relies on Northbound API's, whereas the SDN Controller interfaces with the networking devices via the Southbound API's. Multiple methods may be used for the communication of the data plane devices with the SDN Controller. The most popular one relies on the **OpenFlow**(**OF**) protocol [73], which will be analyzed in subsection 2.4.4.

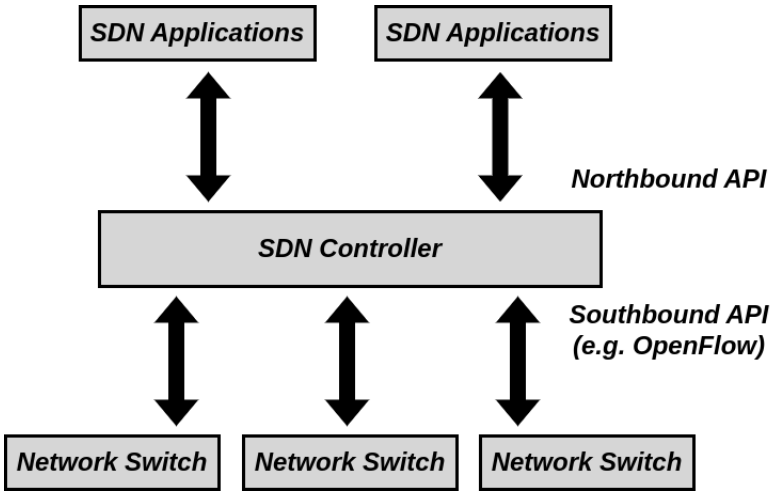

Figure 2.12: Communication between the SDN entities based on Northbound and Southbound API's

The Software-Defined Networking paradigm offers multiple benefits to network administrators. Controlling networks from a central point (i.e. the Controller) boosts network visibility; administrators may effectively monitor their infrastructure and accordingly enforce granular policies across data plane devices. Centralized management may also enhance network reliability by facilitating the automation of complex operations, while resource utilization may be improved by employing advanced traffic engineering algorithms across the network. By separating data and control planes, SDN avoids vendor lock-ins; specific open standards are followed, which enable vendor-agnostic network management.

Despite the increased flexibility, SDN introduces novel challenges. The SDN Controller is a common target of DDoS attacks [74], which aim to render the control plane inaccessible to the networking devices. Therefore, SDN administrators should thoroughly design their networks to safeguard the SDN Controllers by providing multiple instances and/or deploying effective DDoS protection solutions. Furthermore, the latency between the data plane devices and the SDN Controller instances may be significant for latency-sensitive applications, e.g. DDoS attack detection and mitigation that requires high-throughput packet processing.

2.4.4 The OpenFlow (OF) Protocol

OpenFlow (OF) [73] is utilized by the data plane switches to communicate with the SDN Controller (control plane). Switches supporting the OF protocol are called OF switches. Communication between the SDN Controller and the OF switches may be unencrypted or encrypted using the Transport Layer Security (TLS) protocol, which establishes secure channels as depicted in Fig. 2.13.

OF switches handle incoming packets according to the **flow** they belong to; packets are attributed to the same flow if they have equal values in specific header fields (e.g. IP addresses and/or transport layer ports). When a packet is received, OF switches search within their **flow tables** to determine the corresponding **flow entry**, which specifies what action should be performed. If there are no entries associated with a particular flow, hence a switch cannot determine

how to handle a received packet, the packet is forwarded to the SDN Controller using the OF protocol. The Controller subsequently determines the appropriate action that should be performed on the packet and installs a new flow entry at the switch. Thus, subsequent packets belonging to this flow will be handled by the OF switch without communicating with the SDN Controller.

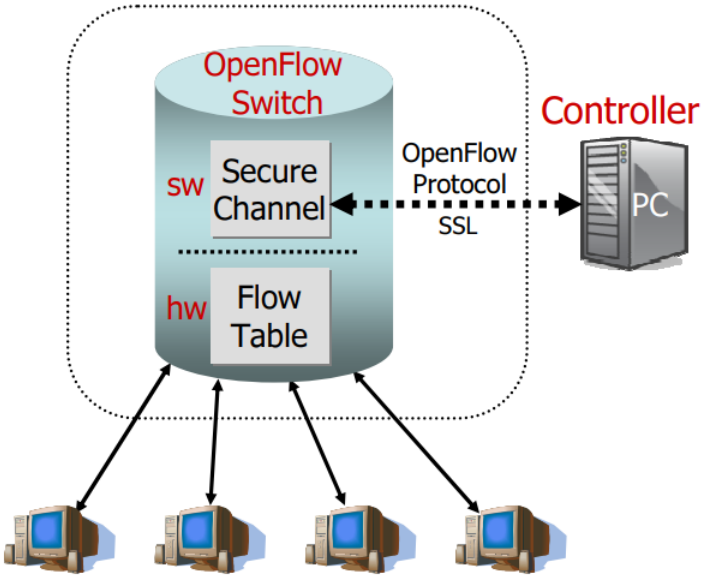

Figure 2.13: The OpenFlow protocol - Communication between the data plane switches and the SDN Controller (figure retrieved from [73])

Each flow entry [75] consists of:

- **Match fields:** The packet header fields constituting a flow entry are called match fields. Header values of ingress packets are compared with those of flow entry match fields; if they are equal, the associated action is performed. Match fields may include the OF switch port that receives the packet, source and destination Medium Access Control (MAC) addresses, source and destination IP addresses, source and destination transport layer ports, VLAN ID's, IP protocol (e.g. UDP, TCP or ICMP) and IP Type of Service (ToS) bits. Flow entries may specify values for all the supported match fields or a subset of them; match fields without a specified value are denoted with a wildcard character (usually "*"). Ingress packets may thus match with multiple flow entries; conflicts arising from multiple matching flow entries for a packet are resolved based on a priority value associated with each flow entry (higher values are prioritized).
- **Counters:** Each time an ingress packet matches a flow entry, one or more associated counters are updated accordingly. Counters may be maintained per flow table, specific flow entries, OF switch ports, etc. Network administrators may utilize these counters to monitor their networks and accordingly program the SDN Controller to take specific actions based on the values of particular counters.
- **Actions:** Flow entries involve actions that should be performed on matching packets. Indicatively, OF switches may (i) forward the ingress packet to a specific OF switch port, (ii)

flood the packet to all OF switch ports, (iii) drop the packet, (iv) modify the packet headers or (v) forward the packet to the SDN Controller for further inspection.

Flow entries can be removed either manually by the SDN administrators or automatically when specific timers expire. Flow entry timers may be: (i) the **idle timeout timer**, which resets each time a packet matches the flow entry and (ii) the **hard timeout timer**, which does not reset each time a matching packet is received and the flow entry is removed when the timer expires.

2.4.5 Programmable Data Planes (PDP's)

The OF protocol initially attracted significant interest within the research community. However, OF design limitations soon became apparent to network administrators. Data plane switches were limited by the OF protocol available features; introducing novel capabilities to future OF versions required time-consuming standardization processes. Furthermore, although network administrators could program their own features at the control plane, the latency introduced by frequent SDN Controller and data plane device communications was restrictive for various time-sensitive applications. Finally, OF focused on packet headers, thus it was not a suitable solution for DPI actions within the data plane.

The aforementioned limitations led to the introduction of Programmable Data Planes (PDP's) [76] as a promising alternative to OF for implementing SDN's. Instead of being limited by the device fixed-function chip capabilities (legacy networks) or the OF supported features (OF-based SDN's), network administrators may leverage on PDP's to directly program the behavior of the forwarding devices. Therefore, received packets are forwarded based on software running within the data plane devices, while communications with external controllers are significantly decreased compared to OF-based approaches. PDP's offer the following advantages:

- **Programmable chips:** SDN administrators directly program the chips of data plane devices. Therefore, experimental features and protocols may be implemented without requirements for any standardization procedures.
- **Cost reduction:** Instead of replacing old networking equipment with newer to support novel features and protocols, SDN administrators may rely on white-box switches, which are able to support new capabilities by simply being reprogrammed.
- **Complexity reduction:** SDN administrators may remove protocols and features, which are not utilized within their networks, to reduce packet header size and the overall complexity of managing their devices.
- **Network visibility:** PDP's may increase the network visibility of SDN administrators by enabling the efficient collection of monitoring metrics within the data plane. Therefore, widely-used legacy network applications may be implemented more effectively within PDP's (e.g. DDoS detection and mitigation).

There are multiple implementations of PDP's. In the following, we describe Programming Protocol-Independent Packet Processors - P4 (subsection 2.4.5.1), eXpress Data Path - XDP

(subsection 2.4.5.2) and Data Plane Development Kit - DPDK (subsection 2.4.5.3), which are important for subsequent chapters of this dissertation.

2.4.5.1 The P4 Language

Programming Protocol-Independent Packet Processors (P4) [12] is a Domain-Specific Language (DSL) used for programming the forwarding logic of data plane switches. P4 programs are compiled and their syntax is similar to that of the C programming language. P4 is commonly employed by the research community for data plane programming.

The devices capable of running the compiled P4 programs are called **P4 targets**. P4 targets include pipelines of blocks mainly responsible for parsing packet fields and subsequently applying specific actions according to packet header and payload values; actions are determined based on match-action tables populated by the P4 target control plane. Some blocks are programmable and their behavior may be fully determined by P4 developers, whereas some blocks are fixed-function and their functionality is predetermined. The programmable blocks of P4 targets as well as their data plane interfaces are specified by the **P4 architecture** [77]. There are various P4 targets, which may either be software-based or hardware-based; depending on the P4 target, specific P4 architectures are supported. Indicatively, a P4 architecture involving both programmable and fixed-function blocks is V1Model, whereas Protocol-Independent Switch Architecture (PISA) consists only of programmable blocks [77].

The process of programming P4 targets is depicted in Fig. 2.14.

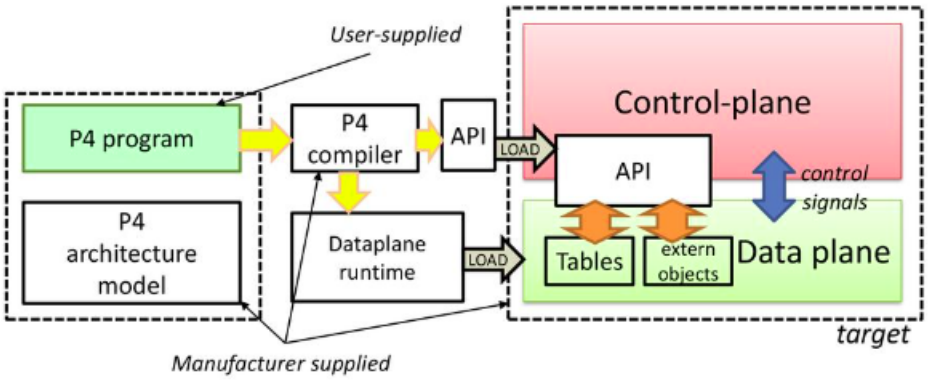

Figure 2.14: The process of programming P4 targets (figure retrieved from [78])

Although the Behavioral Model version 2 (BMv2) [79] is a popular software-based P4 target, the achieved throughput is significantly low, while some P4 features are not supported by the available P4 architecture (i.e. V1Model). These limitations render BMv2 suitable only for learning the P4 language and/or debugging simple programs.

Network engineers usually rely on hardware-based P4 targets. These may include:

- **P4-programmable SmartNIC's:** They are Network Interface Cards (NIC's) supporting the execution of P4 programs. They include a small number of ports, but their throughput is high. Their cost is considerably low. Indicatively, Netronome Agilio CX 25GbE SmartNIC

[80] involves two ports, each supporting speeds up to 25 Gbps, and costs approximately $600 (2023 price).

- **Field Programmable Gate Arrays (FPGA's):** P4 may also be utilized as a high-level programming language for describing FPGA operation; P4 programs are translated into a hardware description language, e.g. Verilog. Compared to P4-programmable SmartNIC's, they include a higher number of high-throughput ports. However, their cost is generally high. Indicatively, NetFPGA-SUME Virtex-7 [81] includes 4 ports, each supporting speeds up to 10 Gbps, and costs $7,000 (2023 price).
- **P4 switches:** Compared to the previous targets, P4 switches include the highest port number and their throughput is significantly increased, but their cost is much higher. Indicatively, Edgecore Wedge 100BF-32X [82] has 32 ports, each supporting speeds up to 100 Gbps, and costs roughly $10,000 (2023 price).

Despite the line-rate packet processing capabilities and the flexibility of programming data plane device chips, there are important constraints [12] that complicate P4 usage and may limit the type of supported applications. P4 is not a general-purpose programming language; loops and pointers are not supported, while variables are of fixed-size. Thus, some applications may not be supported or may require extensive workarounds, e.g. data plane ML inference [83].

2.4.5.2 The eXpress Data Path (XDP) Framework

The eXpress Data Path (XDP) framework [10] is an open-source solution that establishes a programmable data path in the Linux kernel. XDP achieves high throughput packet processing by minimizing the overall number of actions performed on each packet. Received packets are intercepted at the NIC driver level before any memory is allocated to them. This is done via an XDP Hook attached at the earliest possible layer in the kernel network stack; this hook detains ingress packets and delivers them to an extended Berkeley Packet Filter (eBPF) program on a per-packet basis. Processed packets may be either (i) dropped (XDP_DROP action), or (ii) bounced back to the interface they were received (XDP_TX action), or (iii) redirected to a different interface (XDP_REDIRECT action), or (iv) passed to the user space through the network stack (XDP_PASS action), or (v) forwarded to the user space via an optimized AF_XDP socket that bypasses the kernel. An eBPF program [10] may communicate with the user space and/or other programs via eBPF maps available in the system kernel.

Contrary to similar data plane approaches, e.g. the Data Plane Development Kit (DPDK) [11], XDP does not bypass the kernel and does not require specialized hardware, e.g. P4 [12] NIC's. Therefore eBPF programs may benefit from utilities available in the Linux kernel (e.g. TCP/IP libraries) and may adapt to diverse systems if supported by the specific driver version. However, additional effort is required to ensure that XDP operation will not compromise the kernel safety at runtime. To that end, the eBPF verifier is used to guarantee the system safety before the program is loaded. This involves ensuring that there are no unbounded loops present and the eBPF program size [10] is constrained. During execution time, XDP requires checking

that no data are read out of bounds in which case an XDP_ABORTED action is generated.

2.4.5.3 Data Plane Development Kit (DPDK)

The Data Plane Development Kit (DPDK) [11] is an open-source framework that provides utilities for offloading packets from the Linux kernel to the user space for efficient processing. DPDK mainly relies on the following techniques [84, 85]:

- **Fast-path:** High DPDK speeds are achieved by bypassing the kernel network stack and delivering packets from the data plane directly to the user space for processing. Thus, DPDK avoids the latency introduced by the Linux kernel network stack, e.g. context switching (which significantly increases processing delay per received packet) is decreased.
- **Poll Mode Driver** (PMD)**:** Instead of utilizing costly interrupts to deliver ingress packets from the NIC to the CPU, CPU cores poll NIC's for available packets, significantly accelerating packet delivery to DPDK applications.

Despite accomplishing packet processing at remarkably high speeds, DPDK includes some drawbacks. Writing efficient DPDK applications requires getting familiar with a new programming model and a wide number of available libraries. Furthermore, bypassing the Linux kernel renders DPDK applications unable to leverage on widely-used kernel modules (which are available to XDP), e.g. TCP/IP code. Finally, DPDK requires dedicating CPU cores to the running applications, thus they are not shared with other system utilities [85] as in XDP; this may lead to poor CPU utilization.

2.5 Infrastructure Monitoring Tools

Infrastructure monitoring tools are integral for effectively managing computer networks. The available monitoring data enable administrators to verify that their network is operating as expected, eventually facilitating DDoS attack detection and mitigation.

In the following, we describe NetFlow (subsection 2.5.1), sampled Flow - sFlow (subsection 2.5.2) and the collectd system statistics collection daemon (subsection 2.5.3).

2.5.1 NetFlow

NetFlow [86, 87] is a network monitoring protocol developed by Cisco. NetFlow aggregates ingress and egress network traffic into flows, which are groups of packets that share specific header fields; flow-based data are periodically exported towards external collectors for further analysis. Flow definition depends on the NetFlow version. In early NetFlow version, flows were identified by the following seven fields: (i) Monitored device ingress interface, (ii) source IP address, (iii) destination IP address, (iv) IP protocol number (e.g. 1 for ICMP, 6 for TCP and 17 for UDP), (v) UDP/TCP source port (0 for other protocols), (vi) UDP/TCP destination port or ICMP message type and code (0 for other protocols) and (vii) IP ToS.

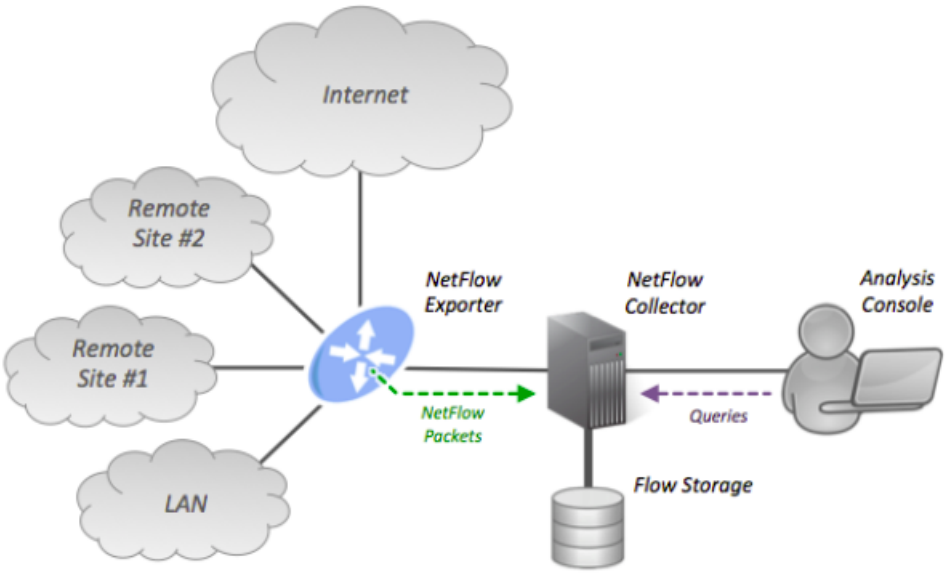

Figure 2.15: The NetFlow architecture for network traffic monitoring (figure retrieved from [87])

The NetFlow architecture for network monitoring is depicted in Fig. 2.15. NetFlow setups usually consist of the following three components:

- **NetFlow Exporter:** Network traffic is aggregated into flows, which are exported towards one or more available NetFlow collectors.
- **NetFlow Collector:** Flow-based monitoring data are collected from multiple network devices and converted into an appropriate format for storage and further processing.
- **NetFlow Analyzer:** Flow-based data are retrieved from NetFlow collectors and analyzed for multiple purposes, e.g. for detecting ongoing attacks.

NetFlow has been widely employed for DDoS attack detection, providing valuable statistics pertaining to packet header fields. However, data sampling is usually required so that NetFlow copes with high-speed DDoS attacks and thus, flows including a small number of packets may be missed. Furthermore, NetFlow does not deliver information about packet payloads, which may be essential for protecting against various attack vectors (e.g. DNS Water Torture attacks).

2.5.2 sFlow

Sampled Flow (sFlow) [88] is a network monitoring protocol suitable for high-speed networks. Packets arriving at network devices (**sFlow agents**) are sampled at configurable rates (i.e. 1 out of n packets). Contrary to NetFlow that considers only packet headers, sFlow samples entire packets. Thus, payload is also included at the cost of increased processing time compared to flow-based monitoring solutions. One or multiple sampled packets are encapsulated within UDP datagrams, which are exported to the sFlow collector for storage and further processing.

2.5.3 System Monitoring with collectd

System monitoring gathers statistics pertaining to the performance of end systems and their running applications. An indicative open-source UNIX-based daemon utilized for system monitoring is **collectd** [89]. Relying on various plugins, collectd is capable of obtaining hardware-based statistics (e.g. CPU, memory and disk information) as well as application statistics (e.g. the number of packets handled by a server). Gathered data are usually streamed to collectors, which may obtain statistics from multiple systems. Indicative collectd plugins are:

- **cpu:** Enables gathering information about the CPU utilization of an end system.
- **memory:** Enables gathering data related to the memory consumption of an end system.
- **dns:** Supports extraction of DNS statistics from a name server, e.g. the total number of (i) received DNS requests, (ii) returned DNS responses or (iii) DNS responses pertaining to specific RCODE values, e.g. NXDOMAIN names.
- **rrdtool:** Enables storing obtained data in Round-Robin Database files (RRDfiles).

The Round-Robin Database tool (RRDtool) [90] may be utilized to retrieve time series data (e.g. CPU utilization) stored within Round-Robin Databases (RRD's) for further processing. RRD's are circular buffers whose memory footprint remains constant over time. Storing new data in RRD's causes the deletion of an equivalent amount of data corresponding to the oldest entries within the database.

2.6 Efficient Data Structures & Algorithms

This section delves into data structures and algorithms, which are utilized in subsequent chapters of the dissertation to efficiently perform tasks related to cyber threat detection and mitigation. Specifically, probabilistic data structures are analyzed in subsection 2.6.1, whereas spelling correctors are briefly discussed in subsection 2.6.2.

2.6.1 Probabilistic Data Structures

Probabilistic data structures are widely employed for Big Data analytics [9]. They are advanced data structures utilized for performing various operations (e.g. membership lookups, frequency estimation and set intersection) in a time and space efficient manner. This efficiency is accomplished by storing elements hashed instead of their original form, which results in the introduction of a configurable percentage of errors. Storing elements hashed also renders probabilistic data structures beneficial for exchanging sensitive data in a privacy-aware format.

In the following, we describe probabilistic data structures, which will be subsequently used for accelerating critical operations related to network security. Specifically, we analyze Bloom Filters - BF's (subsection 2.6.1.1), Cuckoo Filters - CF's (subsection 2.6.1.2) and Count-Min Sketches - CMS's (subsection 2.6.1.3).

2.6.1.1 Bloom Filters (BF's)

Bloom Filters (BF's) [16,91] are probabilistic data structures that perform efficient membership lookups by meeting time and space constraints. They reduce the overall memory consumption by storing elements not in their actual form, but hashed and mapped into a bit array of constant size. Lookups are always accurate for elements stored in the filter (zero False Negatives - FN's) and potentially inaccurate for elements not in the filter (False Positives - FP's). The BF error percentage is affected by the: (i) number of stored elements n, (ii) size of the filter in bits m and (iii) number of hash functions k. The time complexity of element lookups is $O(k)$ and the space complexity is $O(m)$. The FP probability of BF's is approximately given by:

$$(1 - e^{-kn/m})^k$$

Relying on a brute force approach to evaluate all k hash functions would require extensive separate calculations. Instead, the **Double Hashing** technique employs only two hash functions h_1 and h_2 to derive all required k digests without affecting the BF accuracy; the correctness of this approach is formally stated in [91]. The digests d_i for a BF of size m bits are given by:

$$d_i = (h_1 + i \cdot h_2) \mod m, i = 0, 1, ..., k-1$$

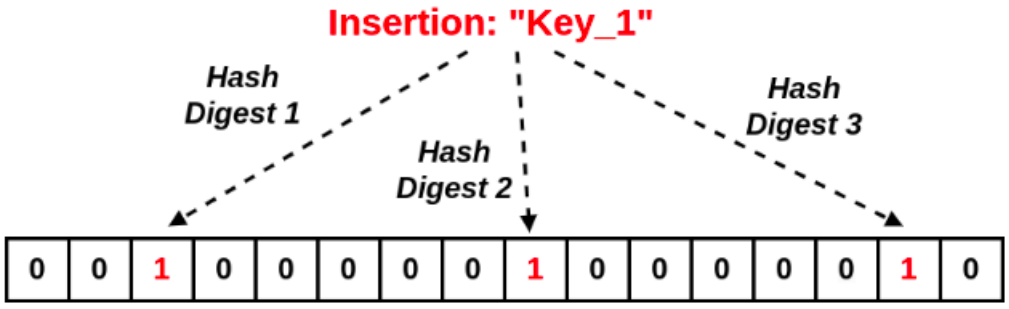

Figure 2.16: Insertion of element "Key_1" in the BF - BF bits determined by hashing element "Key_1" three times are set to 1

Fig. 2.16 depicts the insertion of element "Key_1" in a BF of $m = 16$ bits relying on $k = 3$ hash digests. All BF bits are initially set to 0. After hashing "Key_1" three times, the BF bits determined by the three hash digests are set to 1.

Fig. 2.17 depicts a BF lookup for the element "Key_2", which is not stored within the BF. After hashing "Key_2" three times, the BF bits determined by the corresponding hash digests are inspected. One of the bits equals 0, thus we conclude that "Key_2" is definitely not stored within the BF. Notably, this outcome cannot be wrong because FN's are not possible for BF's; if "Key_2" were in the BF, all bits corresponding to "Key_2" hash digests would equal 1.

Fig. 2.18 depicts a BF lookup for the element "Key_3", which is not stored within the BF. After hashing "Key_3" three times, the corresponding BF bits are inspected. All bits are equal to 1, thus we erroneously conclude that "Key_3" is stored within the BF. FP's are possible for BF's; the bits determined by "Key_3" hash digests have been set to 1 by previous element

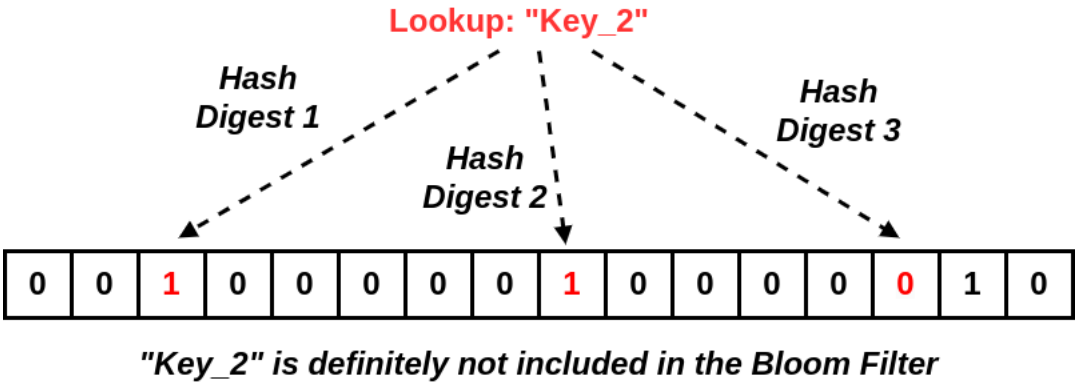

Figure 2.17: BF lookup for element "Key_2" - "Key_2" is definitely not included in the BF because one of the positions corresponding to "Key_2" hash digests is equal to 0

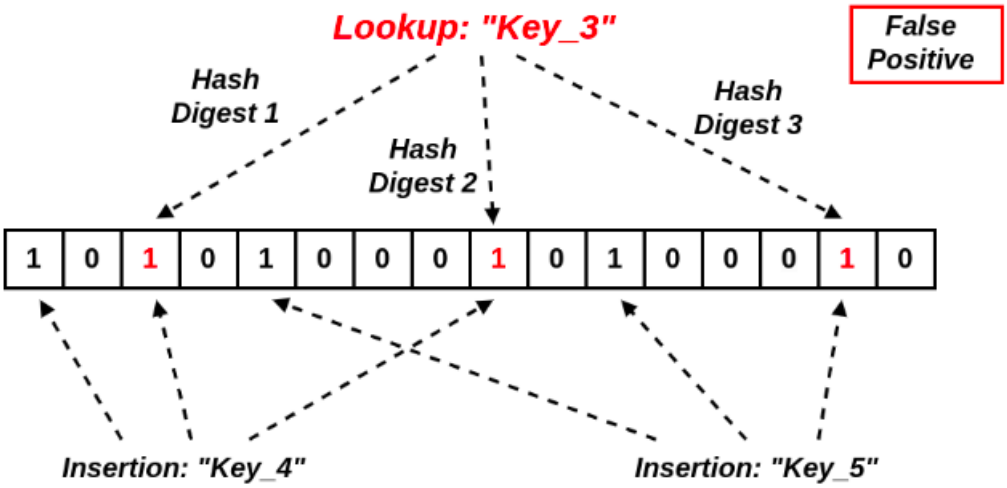

Figure 2.18: Lookup for element "Key_3" erroneously concludes that this element is included in the BF because all bits determined by the three hash digests are set to 1 - This FP is returned because the bits determined by "Key_3" hash digests have been set to 1 by the previous insertion of elements "Key_4" and "Key_5"

insertions (i.e. "Key_4" and "Key_5").

Notably, element deletions are not supported by vanilla BF's. Removing elements by setting their corresponding bits to 0 would introduce FN's because BF positions may be shared by multiple elements.

2.6.1.2 Cuckoo Filters (CF's)

CF's [19] are probabilistic data structures that perform time and space efficient set membership tests. The overall memory consumption is decreased since elements are hashed and inserted in the filter, as fingerprints in the entries of a two-dimensional array. Membership tests are always accurate for elements stored in the filter, but may be inaccurate for those not in the filter. Thus, similarly to BF's [16], FP's are possible, whilst FN's are not. However, contrary to BF's, CF's allow for the efficient deletion of previously inserted elements without introducing space overhead and FN's [91, 92].

CF's are characterized by the number of (i) available buckets m and (ii) multiple fingerprint

entries b per bucket. An element x is hashed into a fingerprint $fgp(x)$ of f bits using the function $fgp()$ [19]. Each element x is assigned to a pair of buckets with indices determined by two additional hashing operations. The index of the first bucket $h_1(x)$ is determined by hashing the element x using a function $hash()$, which may be based on the same algorithm as $fgp()$. The index of the second bucket $h_2(x)$ is determined based on the Partial-key Cuckoo Hashing technique [19]; this involves an XOR operation between $h_1(x)$ and $fgp(x)$ hashed with the function $hash()$:

$$\begin{cases} h_1(x) = hash(x) \\ h_2(x) = h_1(x) \oplus hash(fgp(x)) \end{cases}$$

The computed fingerprint is inserted in either $h_1(x)$ or $h_2(x)$ bucket if space is available. Otherwise, an already inserted fingerprint is randomly selected from one of these buckets and evicted to its alternate bucket if an entry is available. This swapping process continues until empty space is found or a maximum number of allowed evictions is exceeded resulting in insertion failure. Lookups and deletions involve inspecting if an element's fingerprint is in either of its corresponding buckets. The fingerprint size f [19] for an estimated FP ratio ϵ and maximum entries per bucket b is:

$$f \geq \lceil log_2(1/\epsilon) + log_2(2b) \rceil$$

The complexity of CF element insertion is amortized $O(1)$, whilst that of membership testing and deletions is constant, $O(1)$. Apart from enabling element deletion, CF's outperform BF's in terms of lookup time and memory requirements for applications with FP ratio less than 3% and nearly full CF's. BF's remain superior in applications that require frequent insertions of large element sets due to the costly CF eviction process. However, our use cases for privacy-aware zone exchanges (described in Chapter 5) will not require frequent bulk insertions, thus we will opt for CF's due to their low lookup latency and rapid updates for large zones.

2.6.1.3 Count-Min Sketches (CMS's)

Count-Min Sketches (CMS's) [17] are probabilistic data structures that efficiently provide frequency estimations on data stream elements. Instead of maintaining dedicated counters per element, they reduce the overall space requirements by hashing elements and using pools of shared counters. As a result, the actual frequency of elements may be overestimated, while underestimations are not possible. CMS's are characterized by: (i) the number of hash functions k and (ii) the number of counters m per hash function. Increasing the parameters k or m generally reduces the probability of false frequency estimation.

Fig. 2.19 depicts the insertion of element "Key_1" in a CMS that relies on $k = 3$ hash functions and $m = 10$ counters per hash function. After determining the appropriate CMS counters based on the positions returned by hashing "Key_1" three times, the value of each counter is increased by 1.

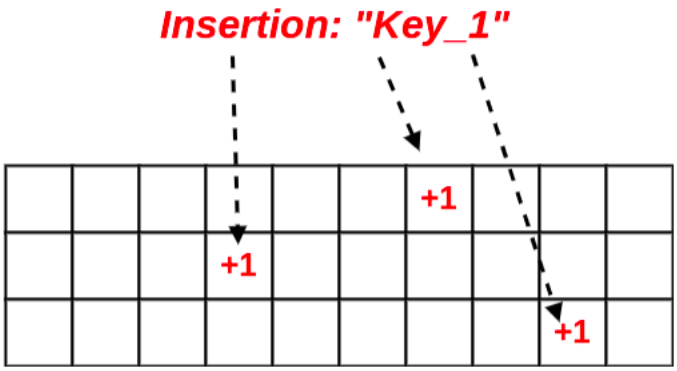

Figure 2.19: Inserting element "Key_1" in the CMS - Counters determined by "Key_1" hashes are increased by 1

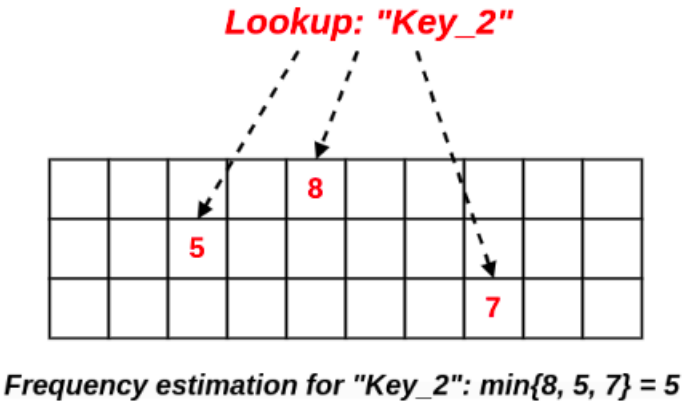

Figure 2.20: Lookup for element "Key_2" in the CMS - The frequency of "Key_2" is estimated as the minimum value of all counters determined by hashing "Key_2"

Fig. 2.20 depicts a lookup for element "Key_2". Initially, the counters corresponding to the inspected element are determined by hashing "Key_2" three times. The frequency estimation for element "Key_2" is determined as the minimum of all counter values.

2.6.2 Spelling Correction Algorithms

Spelling correction algorithms identify typos within given words and enable rectification of spelling mistakes. They are usually based on the **edit distance** between the word under consideration and a dictionary of valid words. A frequently used edit distance metric is the Damerau-Levenshtein [93] that determines the minimum number of character insertions, omissions, substitutions and transpositions to convert a string to another.

Naive spelling correction algorithms compare the given word with every available dictionary term. To reduce the total processing time, Norvig's approach [94] generates beforehand all possible dictionary term variants (resulting from character insertions, omissions, substitutions and transpositions) for edit distances up to a maximum defined value. These variants are stored in the dictionary along with their original terms; this may result into oversized dictionaries. A more efficient approach is followed by the Symmetric delete Spelling correction (**SymSpell**) algorithm [18], which constructs a dictionary containing the actual terms along with their variants

resulting from all possible character omissions up to the considered edit distance. As a result, the overall storage requirements remain low, whilst SymSpell maps the dictionary terms to a hash table to ensure that the average search time complexity is $O(1)$. Notably, search time is independent of the number of terms stored in the dictionary, although it depends on the average length of dictionary terms and the maximum edit distance value under consideration.

2.7 Machine Learning (ML)

This section delves into concepts related to Machine Learning (ML). ML fundamentals are discussed in subsection 2.7.1, whereas ML algorithms utilized in this dissertation are briefly described in subsection 2.7.2. Finally, in subsection 2.7.3 we provide background on eXplainable Artificial Intelligence (XAI) techniques, which may be employed to derive interpretations on ML model decisions.

2.7.1 Machine Learning Fundamentals

Following rule-based approaches to program system behavior may prove hard and extremely time-consuming. Machine Learning (ML) [95, 96] involves algorithms for developing systems, which are capable of performing specific tasks without being provided with explicit instructions. Learning datasets are provided to ML models, which enable them to identify patterns among input **features** and generalize to previously unseen data. Learning datasets are usually split in two portions; one portion is used by ML models to tune their parameters (**training set**), while the other portion is utilized to evaluate model performance (**testing set**).

ML models are broadly categorized in one of the following paradigms according to the approach followed to solve a problem:

- **Supervised learning:** Provided learning datasets include input sampling points and their respective labels, i.e. tags corresponding to the desired model output. ML models configure their parameters by determining the best mapping between input features and output labels that minimizes a given loss function.
- **Unsupervised learning:** Unlabeled learning datasets are provided, i.e. input sampling points without associated labels. ML models configure their parameters by discovering patterns within given data, e.g. clusters of input examples sharing similar properties.
- **Reinforcement learning:** ML algorithms involve agents that learn tasks by receiving feedback from their environment. Specifically, agents aim at maximizing a cumulative reward through a trial and error approach that rewards "good" decisions and penalizes "bad" ones.

ML has revolutionized a wide variety of applications, including cyber threat detection and mitigation, by facilitating the automated identification of useful patterns within the given data instead of rule-based systems, which may be hard to develop. However, ML has also introduced significant challenges. Specifically, ML model performance depends heavily on the volume and quality of the available data; ML applications usually require massive volumes of data, which

should be representative of the targeted problem. Furthermore, ML model training requires substantial amounts of time and considerable computing resources, e.g. multiple GPU's. ML models also introduce decision making errors that may have significant consequences depending on the considered application.

2.7.2 Machine Learning Algorithms

In the following, we briefly describe ML algorithms that will be used in the remainder of this dissertation. Specifically, in subsection 2.7.2.1 we discuss Decision Tree (DT) classifiers, whereas in subsection 2.7.2.2 we describe Random Forest (RF) classifiers. Then, in subsection 2.7.2.3 we provide background on Extremely Randomized Trees (ExtraTrees) and in subsection 2.7.2.4 we describe boosting classifiers, which may involve Adaptive Boosting (AdaBoost), Gradient Boosting (GB) and eXtreme Gradient Boosting (XGBoost). Finally, in subsections 2.7.2.5 and 2.7.2.6, we respectively discuss Naive Bayes Classifiers and Multi-Layer Perceptrons (MLP's) for binary classification.

2.7.2.1 Decision Tree (DT) Classifiers

Decision Tree (DT) classifiers are widely-used supervised learning algorithms. They are non-parametric classifiers, i.e. they do not make any assumptions on the distribution of the provided learning data. DT classifiers are organized in a tree structure; the root and the internal tree nodes correspond to ML model features, whereas the leaves denote model outcomes, i.e. class labels. Similarly to flowcharts, an input example is attributed to a specific class by following if-then-else statements across the tree nodes from the root to the leaves. Training of DT classifiers requires splitting learning datasets into subsets, while maximizing or minimizing a given metric, e.g. the tree Gini index or Entropy respectively.

DT classifiers are simple and fast white-box models; their operation may be fully interpreted by ML model developers contrary to more complex, black-box models. However, they are prone to overfitting, which becomes more evident as the tree depth and number of model features increase.

2.7.2.2 Random Forest (RF) Classifiers

Random Forest (RF) classifiers are non-parametric, supervised learning ML algorithms. Relying on ensemble learning, they aggregate the predictions of multiple DT classifiers, eventually decreasing the overall bias and overfitting. During the learning phase, RF classifiers ensure that the correlation among individual predictors is low. This is accomplished mainly by (i) employing bootstrap aggregation (bagging), which involves randomly subsampling learning datasets with data replacement and (ii) feature randomness, which involves selecting random subsets of the available features. During the testing phase, input examples are assigned to the class predicted by the majority of the individual DT's. RF's are more accurate than DT's, but they are

more resource-intensive, whereas their operation is not interpretable, i.e. they are black-box models.

2.7.2.3 Extremely Randomized Trees (ExtraTrees)

Extremely Randomized Trees (ExtraTrees) are supervised learning algorithms relying on ensemble learning techniques for categorizing input examples. Similarly to RF classifiers, ExtraTrees combine multiple DT's. However, ExtraTrees are generally faster than RF's and individual DT's are trained on dataset subsets, which are sampled without data replacement. Therefore, each separate predictor is trained on a unique portion of the dataset. This ensures that the individual DT's are uncorrelated.

2.7.2.4 Boosting Classifiers

Boosting classifiers [97] are non-parametric, supervised learning algorithms. They rely on ensemble learning techniques that combine multiple weak learners (e.g. DT classifiers) to build stronger ones. Specifically, boosting classifiers require sequentially developing ML models based on the output of previously trained, weaker learning models. The developed models are aggregated to a single one to increase classification performance.

Multiple ML algorithms rely on boosting, such as Adaptive Boosting (AdaBoost), Gradient Boosting (GB) and eXtreme Gradient Boosting (XGBoost). Indicatively, AdaBoost combines individual predictors based on a weighted linear function that assigns larger weights to stronger learners. Individual learners of AdaBoost are usually decision stumps, i.e. DT classifiers including a single input feature. Similarly to RF's, boosting classifiers are also black-box models.

2.7.2.5 Naive Bayes Classifiers for Binary Classification

Naive Bayes Classifiers [98] are supervised ML algorithms that assume conditional independence among features given the class of the output. This assumption renders Naive Bayes easy to implement and fast in evaluating inputs, while less time and data are required for training compared to other ML algorithms.

Training a Naive Bayes Classifier to predict the class y of input $[x_1, ..., x_m]$ given m features X_i requires estimating two types of Maximum Likelihood Estimations (MLE's):

- MLE's for prior probabilities, i.e. $\widehat{P}(y)$, calculated by dividing the number of training examples belonging to class y with the total number of examples in the training dataset.
- MLE's for conditional probabilities of each feature X_i given a class y, i.e. $\widehat{P}(X_i = x|y)$, calculated by dividing the number of training examples belonging to class y and their feature X_i equals x with the total number of training examples belonging to class y.

Then, a new test instance $[x_1^{new}, ..., x_m^{new}]$ is assigned by Naive Bayes to class y^{new} based on:

$$y^{new} \leftarrow \underset{y}{\operatorname{argmax}}\, \widehat{P}(y) \prod_{i=1}^{m} \widehat{P}(x_i^{new}|y)$$

2.7.2.6 Multi-Layer Perceptrons (MLP's) for Classification

MLP's [96] are feedforward neural networks utilized for supervised learning purposes. They include multiple neurons, which are organized in layers, specifically an input layer, an output layer and one or more internal layers between them (called hidden layers). Neurons of the input layer and the hidden layers are connected to each neuron of the subsequent layer, whereas the outcomes of output layer neurons are used to derive the class of the provided input examples.

Training of MLP's relies on the backpropagation algorithm, which involves tuning (i) the weights of the links between neurons, i.e. decimal numbers that specify the amount of the neuron outputs that will be transferred to the next MLP layers and (ii) the bias parameter of each neuron, which is a constant numeric value. The output of each neuron is determined by adding the bias parameter to the weighted sum of neuron inputs and subsequently feeding the result to a non-linear activation function, e.g. ReLU or sigmoid. Activation functions enable MLP's to capture complex relationships between input data and their provided labeled outputs, whereas the model is capable of generalizing to previously unseen data.

Similarly to RF classifiers, boosting classifiers and ExtraTrees, MLP's are black-box models. Thus, their operation is not inherently interpretable unless XAI methods are used.

2.7.3 eXplainable Artificial Intelligence (XAI)

ML algorithms have been widely adopted for processing massive amounts of data in a highly efficient manner for image processing, cybersecurity, healthcare, online education, etc. During the last decades, practitioners have been focusing on increasing the predictive capabilities of ML models. Therefore, simple and intrinsically explainable models have been replaced by complex networks that achieve higher performance. However, these complicated, black-box models are not interpretable, hence they introduce significant issues to their developers, their users and legal entities who supervise their deployment. Developers are incapable of understanding their models to debug them and assert their intended operation, while users cannot receive justifications on model decisions made on their data. Finally, legal regulators are unable to ensure that models deployed within critical infrastructures comply with GDPR [99] or equivalent mandates.

Driven by the aforementioned limitations, **eXplainable Artificial Intelligence (XAI)** techniques have been proposed to provide interpretations on the operation of ML models, including deep neural networks. Although various techniques have been proposed, the most appealing XAI algorithms are those categorized as post-hoc and model-agnostic. Such algorithms are applied after the model is trained (post-hoc) and they are independent of selected models (model-agnostic), therefore they constitute a promising category of XAI algorithms. Interpretations may be *global*, detailing the model behavior on multiple sample points, and *local*, reporting how models make classification decisions for specific input examples.

In subsection 2.7.3.1 we briefly describe **SHapley Additive exPlanation (SHAP)**, which is a widely-used post-hoc and model-agnostic XAI method that provides both global and local model interpretations.

2.7.3.1 SHapley Additive exPlanation (SHAP)

SHAP [13,15] is a model-agnostic, post-hoc XAI method related to cooperative game theory. In cooperative games, players collaborate to achieve a pay-off, which is subsequently split based on participant contributions. Accordingly, features are considered as participants that tune a classifier and subsequently SHAP determines feature importance by estimating the effect of specific features on classification decisions when these features are present and absent.

SHAP delivers global and local explanations [14] on ML model decisions, whereas various visualization tools facilitate interpretations, e.g. summary plots, dependence plots and force plots. Model-agnostic SHAP is typically based on the KernelExplainer [100] method; this approximates feature importance via a weighted linear regression model applied to input instances (sample points). SHAP time complexity mainly depends on the dataset size. Enabling execution within reasonable time frames may require clustering and/or subsampling a given dataset. This process extracts the eXplainability Background Instances (XBI's) used for tuning SHAP values and eXplainability Test Instances (XTI's) utilized for generalizing model interpretations.

Chapter 3

Water Torture Attack Protection based on Efficient Algorithms & Probabilistic Data Structures in the User Space

This chapter[1] presents a privacy-preserving schema between Authoritative and Recursive DNS Servers (Recursors) for the efficient detection and collaborative mitigation of DNS Water Torture attacks in cloud environments. Monitoring data are harvested from the victim premises (Authoritative DNS Server and data center switches) to detect anomalies with DNS requester IP addresses classified as legitimate or suspicious. Subsequently, requests are forwarded or redirected for refined inspection to a filtering mechanism. Mitigation may be offered as a service either on-premises or via cloud scrubbing infrastructures.

The proposed schema leverages on probabilistic data structures (Bloom Filters - BF's, Count-Min Sketches - CMS's) and related algorithms (SymSpell) to meet time, space and privacy constraints. Notably, BF's are employed to map Resource Records (RR's) of large DNS zones in a memory-efficient manner; rapid name lookups are possible with zero False Negatives (FN's) and tolerable False Positives (FP's). Our approach is tested via a proof of concept setup based on traces generated from publicly available DNS traffic datasets.

3.1 Motivation

As briefly mentioned in Chapter 2, DNS DDoS attacks present an ever-increasing threat to the operation of the Internet, whereas the severe impact of DNS outages became evident during the Dyn DDoS attack (see Chapter 1). According to [3], Dyn servers were overwhelmed by attack traffic that reached the immense volume of 1.2 Tbps. This was forwarded by roughly 100,000 IoT devices infected with the Mirai malware [101] presumably utilizing DNS Water Torture and Generic Routing Encapsulation (GRE) Flood as attack vectors [102]. During the attack, Internet services of major companies, e.g. Amazon, Twitter and Netflix [101] were severely affected.

[1] This work is included in the proceedings of the 8th IEEE International Conference on Cloud Networking (IEEE CloudNet 2019)

DNS Water Torture [102] overwhelms the computing resources of Authoritative DNS Servers with invalid Fully Qualified Domain Names (FQDN's) not contained in the victim's zone (see subsection 2.3.1.2). These are randomly generated so that a name is never repeated, bypassing the DNS caches of caching Recursors. Thus, the attack traffic is forwarded to victims. Note that another commonly used DDoS attack vector is DNS Amplification (see subsection 2.3.1.3) that involves unsolicited amplified responses by exploiting vulnerable Recursors, including Open Resolvers, which are openly available to every Internet user. This vector represents a volumetric attack that saturates network links and is commonly mitigated via traffic blackholing or redirection to scrubbing services. This chapter focuses on protecting Authoritative DNS Servers, which are critical Internet resources, in a cost-effective and efficient manner against Water Torture attacks and not volumetric DDoS attacks.

We present a privacy-preserving schema among collaborating Authoritative DNS Servers and Recursors for the efficient detection and mitigation of DNS Water Torture attacks. Our schema meets time and space constraints both in detection and mitigation stages by leveraging on probabilistic data structures, i.e. Bloom Filters (BF's), Count-Min Sketches (CMS's) and related algorithms, i.e. SymSpell. Monitoring data are harvested from the victim infrastructure; these enable a classifier to categorize requester IP addresses according to the validity and frequency of requested names. When a Water Torture attack is detected, traffic from attack sources is redirected for further inspection to a filtering appliance. Mitigation may be offered as a service either on-premises or via cloud scrubbing services. Moreover, sources of suspicious traffic, e.g. Recursors, are notified by the victim; these may deploy filtering instances to mitigate attacks closer to their origins and therefore deliver high quality services to their users.

We are mainly based on the idea of employing BF's as name whitelists and map the RR's of large DNS zones in a memory-efficient manner. BF's ensure fast element lookups with zero FN's and tolerable FP's. Moreover, their privacy-preserving nature supports mitigation enforcement closer to the attackers without disclosing the contents of zones to third parties. Thus, they are suitable for the mitigation of DNS DDoS attacks in cloud infrastructures that impose particular time, memory and privacy requirements.

The remainder of this chapter is structured as follows: In Section 3.2, we discuss works related to our approach; in Section 3.3, we provide a high-level overview of our schema and analyze its design features; in Section 3.4, we discuss implementation details pertaining to our proposed architectural components; in Section 3.5, we present the results of our experiments. Finally, in Section 3.6, we summarize our work.

3.2 Related Work & Contributions

There are several approaches reported in the literature, e.g. [103,104] that suggest using BF's for whitelist/blacklist generation and CMS's for frequency statistics in DDoS attack detection and mitigation. Our work builds upon the privacy-preserving properties and space-time efficiency of these probabilistic data structures to develop a schema for the protection against DNS Water

Torture attacks, which may be deployed in cloud infrastructures. In the sequel, we refer to related work pertaining to handling DNS DDoS attacks.

In [5] an approach is proposed to detect DNS Water Torture attacks on Authoritative DNS Servers based on the frequency and statistical features of the requested names. This approach includes a model that analyzes the requested names based on a learning phase that involves training delays. In contrast, our proposed schema depends on the actual zone RR's to validate a name using efficient data structures (BF's) for filtering suspicious traffic.

In [105] a BF-based method is proposed to mitigate DNS Amplification volumetric attacks. The main idea is that a legitimate response corresponds to each outgoing request. Thus, BF's are used to efficiently store summaries of DNS requests and subsequently drop unsolicited responses. Our BF-based method protects Authoritative DNS Servers from Water Torture attacks, which are not volumetric in their nature.

In [106], an IBM US Patent Application outlines methods for filtering DNS requests in Recursors using whitelists/blacklists. This approach monitors the DNS response time of Authoritative DNS Servers and Recursors to pinpoint those requiring protection. Similarly to our schema, the authors propose the utilization of efficient data structures, e.g. BF's, to store one or more valid names selectively prefetched from Authoritative DNS Servers and, accordingly, filter offensive traffic. Furthermore, they do not support their proposal with experimental results. In contrast, our schema uses BF's to map the entire zone contents of Authoritative DNS Servers and is validated with experiments tailored to the requirements of cloud services. Moreover, our approach includes a fine-grained classification process that profiles requester IP's to identify potentially malicious users. This enables enforcing DNS DDoS mitigation closer to the attack sources via a collaborative mechanism amongst Authoritative DNS Servers and Recursors.

3.3 Proposed Schema: High-Level Overview & Design Features

This section provides a general description and lists the design principles of the proposed architecture.

3.3.1 High-Level Description & Features

A high-level description of our schema is depicted in Fig. 3.1. The Authoritative DNS Server receives requests from Recursors for FQDN's within its zones. Data packets are sampled from an SDN-enabled switch (SDN Switch) and DNS metrics are collected from the Authoritative DNS Server. These are fed to an anomaly detection process to identify abnormal DNS activity and classify the corresponding requester IP addresses. DNS messages from suspicious IP addresses are redirected to the DNS Firewall (DFW), thus offloading the Authoritative DNS Server. Instances of this component may be deployed either on-premise or as cloud scrubbing services. Moreover, Recursors forwarding suspicious traffic are notified. These are asked to collaborate by filtering requests in their facilities, presumably closer to the attack sources, based on the contents of the victim zones.

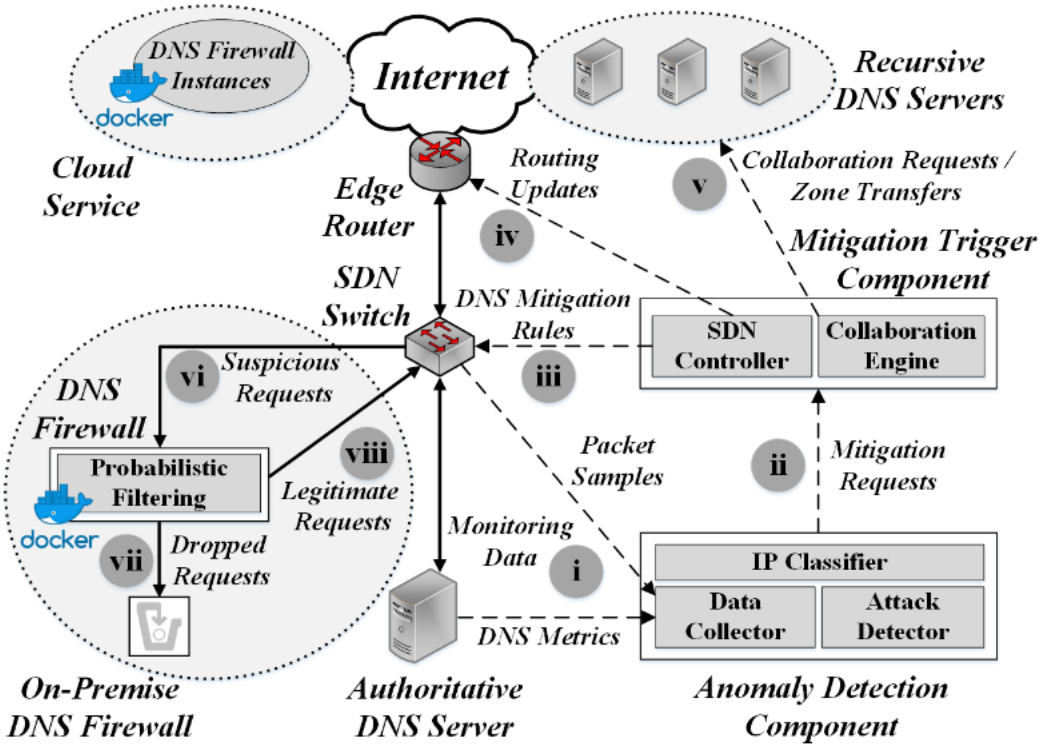

Figure 3.1: Baseline design of the proposed schema

The main design features of the proposed schema are:

- **Lightweight DNS monitoring:** Packet sampling based on sFlow and lightweight statistics collection agents (i.e. collectd [89]) are used to efficiently harvest DNS-related data. Thus, our schema may adapt to great traffic volumes without introducing considerable overhead to the monitoring infrastructure.
- **Fine-grained classification of requester IP addresses based on efficient data structures and algorithms:** Our classification schema leverages on probabilistic data structures (BF's [16], CMS's [17]), along with a deterministic spelling correction algorithm (SymSpell [18]) to classify DNS requester IP addresses as legitimate or suspicious. Requests from suspicious IP addresses are redirected for refined inspection, whereas legitimate traffic is forwarded directly to the Authoritative DNS Server.
- **Space and time efficient filtering of DNS requests:** We utilized BF's for rapid name lookups in a memory-effective manner. Initially, the FQDN's for each Authoritative DNS Server zone are stored in a BF. Subsequent requests about non-existent names are dropped, whilst valid ones are forwarded to the Authoritative DNS Server. BF's ensure that only invalid requests are dropped during filtering (zero FN probability), while FP's may occur with tolerable and configurable probability.
- **Modular DNS Firewall:** Leveraging on the privacy and space-time efficiency properties of BF's, we offer a mitigation solution suitable to be provided as a Virtual Network Function (VNF) [107]. DFW instances may be dynamically deployed at any point across the

attack path within or without the victim's network, while BF's safeguard privacy concerns by hashing DNS zone contents. In particular, DFW instances may be embedded within Authoritative DNS Servers, Recursors or offered as a service within cloud scrubbing infrastructures. Thus, depending on the magnitude of the attack, the mitigation process may be orchestrated via a collaborative schema and pushed closer to the sources of the attack.

3.3.2 Architectural Components - Overview

Our overall architecture consists of three components: (i) the Anomaly Detection Component (ADC), (ii) the Mitigation Trigger Component (MTC) and (iii) the DNS Firewall (DFW). In the following, we outline the functionality offered by each component.

3.3.2.1 Anomaly Detection Component (ADC)

This component performs anomaly detection and classification pertaining to the DNS service. Decisions are based on monitoring data harvested from the SDN Switch and the Authoritative DNS Server. The ADC consists of: (i) the Data Collector (DC), (ii) the Attack Detector (AD) and (iii) the IP Classifier (IPC).

The DC gathers non-sampled DNS metrics from the Authoritative DNS Server and packet samples from the SDN Switch. It isolates DNS samples and extracts specific attributes from the packet headers and payload. The AD retrieves from the DC DNS metrics related to the detection of DNS anomalies. The IPC categorizes DNS requests and related IP addresses as legitimate or suspicious. Classification is conducted in two levels:

- The first level identifies requested FQDN's as valid (present in the corresponding zone) or invalid. Invalid names may occur due to typos, requests for obsolete names or malicious intent, e.g. requests for names generated by Domain Generation Algorithms (DGA's). Appropriate counters for typos and DGA names per requester IP are kept in CMS's.
- The second level employs operator-defined thresholds on the aforementioned counters to classify DNS requester IP addresses as legitimate or suspicious.

3.3.2.2 Mitigation Trigger Component (MTC)

This component enforces the necessary mitigation actions for DNS-related anomalies. The MTC receives Mitigation Requests containing the identified suspicious IP addresses from the ADC. Suspicious requests are redirected to the DFW for refined inspection; this may be located either on-premises or in cloud scrubbing infrastructures. The SDN Controller installs appropriate flow rules (SDN Switch) to divert suspicious requests to the on-premise DFW's or triggers routing updates (Edge Router) to redirect DNS traffic. Moreover, the MTC initiates collaboration requests towards Recursors that forward suspicious traffic. Those willing to filter offensive traffic receive the DNS zones in a privacy-aware format (i.e. BF's) via the Collaboration Engine.

3.3.2.3 DNS Firewall (DFW)

The purpose of this component is to handle suspicious DNS requests based on a BF containing the Authoritative DNS Server zone FQDN's. Requests pertaining to names contained in the BF are forwarded to the Authoritative DNS Server. The zero FN probability ensures that all legitimate requests will reach the Authoritative DNS Server, while requests failing the membership test are dropped. However, a small percentage of invalid requests may be forwarded to the Authoritative DNS Server due to the non-zero FP probability of the BF. Depending on the severity of the attack and the Authoritative DNS Server capabilities, administrators may fine tune the BF size to sustain a tolerable probability of FP's.

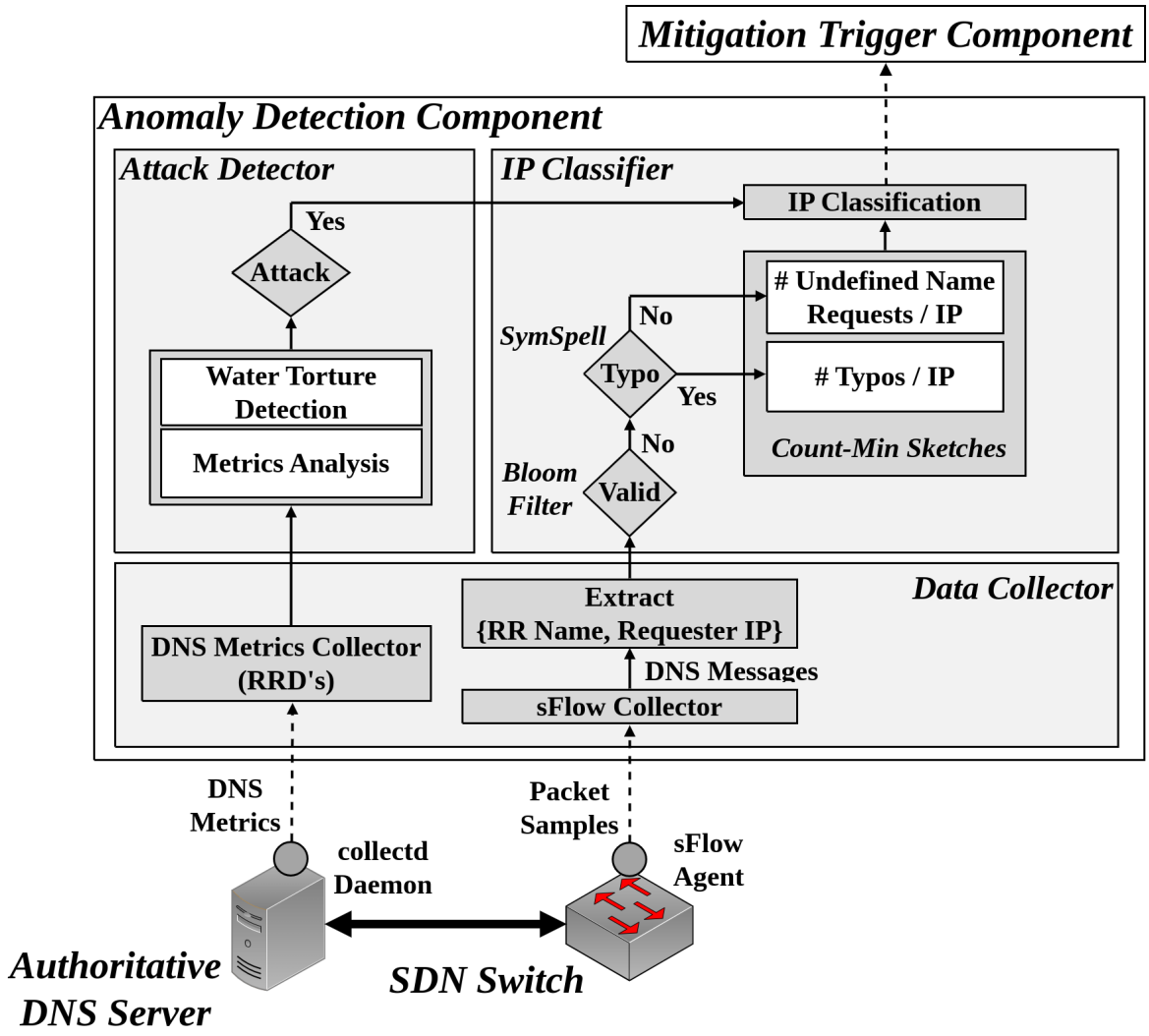

Figure 3.2: The Anomaly Detection Component

3.4 Architectural Components & Implementation Details

This section provides details pertaining to the implementation of the Anomaly Detection Component (ADC), the Mitigation Trigger Component (MTC) and the DNS Firewall (DFW).

3.4.1 Anomaly Detection Component (ADC)

In this part we discuss implementation details of the ADC. This component (Fig. 3.2) was implemented in Python and consists of the following modules:

3.4.1.1 Data Collector (DC)

The DC harvests monitoring data to detect DNS anomalies and classify requester IP addresses. Monitoring data consist of (i) non-sampled DNS metrics gathered from the Authoritative DNS Server using collectd and (ii) DNS messages from the SDN Switch obtained through packet sampling via sFlow. The aforementioned tools are lightweight and thus, suitable for high-traffic networks.

DNS metrics include the total number of DNS responses with (i) NXDOMAIN response code (RCODE) and (ii) any RCODE in general. These are delivered to the DNS Metrics Collector and stored in round-robin databases, which ensure that stale information is efficiently removed, whilst storage requirements remain constant over time.

Packet samples are collected from the SDN Switch. We extract the requester IP address from the IP header and the FQDN first label (prefix) from the DNS payload. Sampled DNS messages may either be requests or responses; our mechanism determines the requester IP address from the corresponding layer 3 field. As the sFlow collector, we used sflowtool [108] and selected sampling rates based on the interface speed as proposed in [109].

3.4.1.2 Attack Detector (AD)

The AD processes information from the DC to detect Water Torture attacks. Collected DNS metrics are retrieved in configurable time intervals via RRDtool [90], a standard tool for RRD files. An attack is detected if the ratio of NXDOMAIN responses to the total number of DNS responses, evaluated in intervals of 5 sec, violates a predefined threshold. This interval accomplishes a satisfactory trade-off between false and delayed decisions. Those based on short intervals may be affected by instant fluctuations thus, benign IP addresses may be misclassified; long intervals may delay the detection of suspicious IP addresses.

3.4.1.3 IP Classifier (IPC)

The IPC classifies requester IP addresses as legitimate or suspicious. The DC feeds the extracted prefix to a BF that contains the valid FQDN prefixes of the Authoritative DNS Server zones. This determines the validity of prefixes without misclassifying valid ones (recall that FP's are possible).

Identified invalid prefixes are further inspected to determine those resembling valid ones. To that end, we employ the SymSpell algorithm to efficiently calculate edit distances in constant time. The SymSpell dictionary is derived using all zone prefixes and their variants resulting from character omissions up to a maximum edit distance value. This value depends on the tolerance for typos. We selected a maximum edit distance of 3 characters assuming that common spelling

mistakes are within this value. Our approach monitors typos as an additional functionality to detect unintentionally invalid requests. Milder classification thresholds may be defined for them than for users forwarding undefined names.

After evaluating the validity of requested prefixes, CMS's are employed to estimate the frequency of (i) typos and (ii) undefined names per requester IP. The estimations of the two CMS's are examined to classify requester IP addresses as legitimate or suspicious. An IP is classified as suspicious if one of the two estimations calculated over a specific time window (e.g. 5 sec) violates a predefined threshold. Traffic originating from this IP is redirected to the DFW. Note that redirection of IP addresses takes place only when a DNS Water Torture attack has been detected. When the time window is over, the sketch contents are cleared. CMS's may overestimate frequencies. Thus, we selected CMS parameters as in [17] to ensure a satisfactory trade-off between false frequency estimations and consumed memory.

3.4.2 Mitigation Trigger Component (MTC)

This component receives Mitigation Requests from the ADC, redirects suspicious traffic to protect the Authoritative DNS Server and optionally sends collaboration requests to Recursors. In general, DNS requests from Water Torture sources are redirected to DNS Firewall (DFW) instances for further inspection via the SDN Switch and/or the Edge Router. The former diverts suspicious traffic to an on-premise infrastructure, while the latter is used to redirect DNS traffic to a cloud scrubbing facility.

The MTC is mainly based on the Ryu SDN Controller [110]; a Python-based framework that provides various built-in components and a convenient REST API. We utilized the OpenFlow (OF) [73] capabilities and BGP Speaker of Ryu to interface both with the SDN Switch and the Edge Router (Fig. 3.1). The redirection rules may consist of OF rules, BGP routes or BGP FlowSpec rules [50]. These should have higher priority than the default used to forward traffic from legitimate sources. Administrators may set the OF idle and hard timeouts based on experience from the duration of previous DDoS attacks.

It should be noted that modern DDoS attacks usually involve a great number of attack sources. By installing mitigation rules per attack source, the capacity (i.e. Ternary Content-Addressable Memory - TCAM entries) of the SDN Switch and/or the Edge Router may be exceeded. Thus, administrators may consider aggregating entire IP prefixes [111] at the expense of potentially redirecting legitimate IP addresses.

3.4.3 DNS Firewall (DFW)

The DFW is mainly based on a BF containing the FQDN's of the Authoritative DNS Server zone RR's. It receives incoming DNS requests and extracts the requested FQDN from the question section in the payload. Subsequently, the BF verifies if this name is valid or not. If it is valid or perceived as valid because of an FP, the message is forwarded to the Authoritative DNS Server. Otherwise, the message is discarded without further notice. The DFW operates transparently, i.e.

it forwards presumably valid DNS messages to the Authoritative DNS Server without modifying the IP header.

The DFW is packaged in a Docker container and may be deployed on demand in various generic infrastructures. In Fig. 3.3 we illustrate a container-based NFV Infrastructure (NFVI), which is based on adapting the Network Functions Virtualization (NFV) framework [107] to deliver DNS-related services.

To that end, we employ the Kubernetes Container Management Stack that offers a unified management system for Docker containers on Linux-based hosts as in [112]. An external API may be used to dynamically provision, scale and terminate instances of the DFW for a specific DNS zone. DFW instances are prepended with a load balancer to cope with the immense DNS traffic during DDoS attacks.

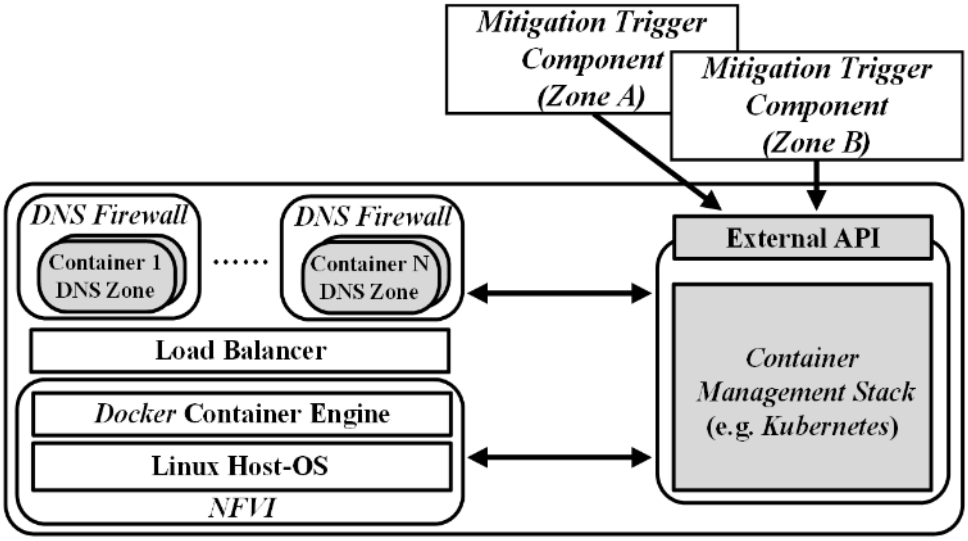

Figure 3.3: DNS Firewall as a VNF

3.5 Evaluation

In this section we demonstrate the appropriateness of BF's for mapping DNS zones and assess the proposed schema.

3.5.1 Application of Bloom Filters (BF's) in DNS

Table 3.1 exhibits the appropriate BF sizes that achieve a targeted FP probability for various DNS zone sizes. We also depict indicative zones for each size and the overall memory required to store their names plaintext. These zones were retrieved in July 2019 from [113,114] except for "ntua.gr" that was available via AXFR within our NTUA campus. This table proves that BF's can reliably map RR's of large DNS zones in a memory-efficient manner with a tolerable FP probability. Notably, a filter of 16.95 MB may be used for the "ru" zone with an FP probability of 0.01%, i.e. 1 in 10,000 invalid names may be considered valid. Note that the "ru" zone is included in the top 10 zone sizes [7]. Thus, BF's may meet the memory requirements of cloud

services.

The FP percentages in Table 3.1 were evaluated using the BF FP approximation formula in [16, 91] (actual FP percentages are slightly worse for the given parameters). Recall that BF lookup times increase with the number of used hash functions. The exact number of hash functions, i.e. 5, was selected as an acceptable trade-off between BF size and lookup time [91]. More hash functions will not reduce the BF memory requirements significantly, while unnecessary increases in BF lookup time will be introduced.

FQDN Entries (Indicative Zone)	**Plaintext Zone Size**	**Bloom Filter Sizes for Specified False Positive Percentage**		
		1%	0.1%	0.01%
8,303 (ntua.gr)	58.59 KB	9.98 KB	17.52 KB	29.37 KB
1,470,834 (se)	21.61 MB	1.73 MB	3.03 MB	5.08 MB
4,905,628 (ru)	58.93 MB	5.76 MB	10.11 MB	16.95 MB

Table 3.1: Application of Bloom Filters in DNS

3.5.2 Evaluation of the Proposed Schema

In this subsection we evaluate our proposed schema. Specifically, we describe the utilized dataset (subsection 3.5.2.1), we demonstrate the effectiveness of the ADC (subsection 3.5.2.2) and assess our DFW (subsection 3.5.2.3).

3.5.2.1 Attack & Legitimate Traces

To the best of our knowledge, there are no publicly available traces of DNS Water Torture attacks. Thus, we created our Attack Traces based on available DNS Amplification attack traces to maintain the statistical properties of actual DDoS attacks. Specifically, we preserved the number and order of packets per IP from the "Booter 4" dataset [22] and modified the message payload to contain generated names between 4 and 20 characters. These were randomly selected from the Latin alphabet, digits 0-9 and hyphens. We did not insert typos in our traces as SymSpell is a deterministic algorithm that always detects typos. However, we included SymSpell in our experimental pipeline to account for potential time delays.

Legitimate Traces were based on data provided by [115]. These contained DNS requests from Google's Public Recursors towards an Authoritative DNS Server located at SURF, the Dutch National Research and Education Network (NREN). The selected trace (December 2017) includes traffic from 1,851 unique sources. Similarly to the Attack Traces above, we maintained the number and order of packets per IP and reconstructed the payload with names uniformly chosen from the "ntua.gr" zone.

3.5.2.2 Anomaly Detection Component (ADC)

In the following we report on the effectiveness of the ADC in classifying requester IP addresses via a proof of concept setup based on Fig. 3.1. We utilized Tcpreplay [116] to feed the traces to the Authoritative DNS Server through the SDN Switch with the following constant rates: (i)

100 Kpps for the Attack Traces and (ii) 3 Kpps for the Legitimate Traces. The Authoritative DNS Server is based on BIND [25], whilst the SDN Switch is based on the Open vSwitch (OvS) virtualization software [117] that supports the OF protocol. We constructed the zone file of the Authoritative DNS Server using 8,303 names collected from hosts under the zone "ntua.gr".

The rate of the Legitimate Traces was based on statistics of the "ch" Top-Level Domain (TLD) Authoritative DNS Servers [118]. The primary server (a.nic.ch) received approximately 1.5 Kpps of traffic (July 2019). Due to the excessive size of the "ch" TLD (about 2 million FQDN's), we consider this rate an indicative value for the majority of TLD and second-level domain name servers. We experimented with the double rate of 3 Kpps to guarantee the effectiveness of our mechanism on lower traffic rates. Regarding the Attack Traces, we selected a rate that is sufficient to downgrade the service provided by the Authoritative DNS Server.

The forwarded traces (Attack and Legitimate) were sampled at the SDN Switch using various sampling rates. The AD detected a Water Torture attack when the ratio of NXDOMAIN responses to the total number of DNS responses was more than 15% (available from "ntua.gr" Authoritative DNS Servers). During time windows of 5 sec, requester IP addresses were classified by the IPC as suspicious if 2 invalid requests were sampled from them. In our experiments we redirected traffic from DNS Water Torture attack sources to an on-premise infrastructure with 5 DFW instances.

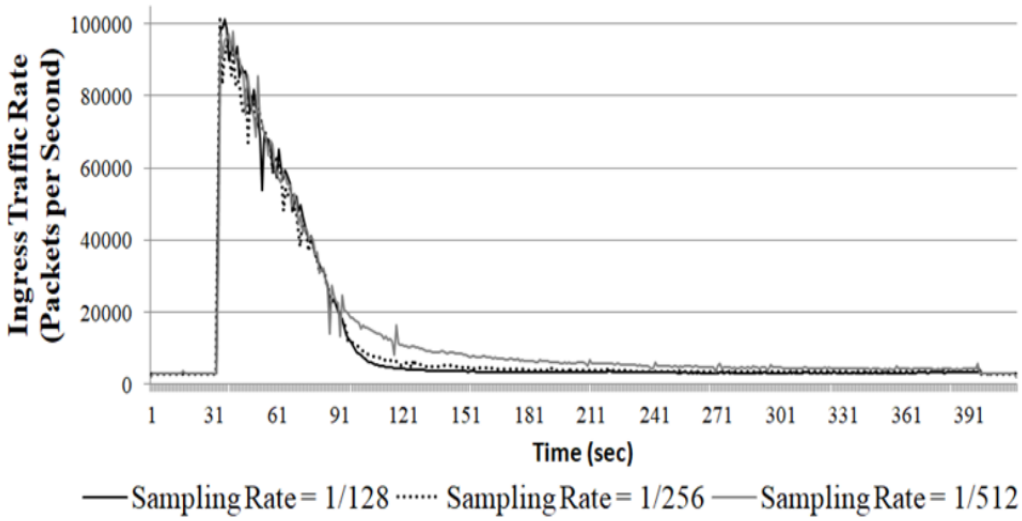

Figure 3.4: Authoritative DNS Server ingress traffic rate for various sampling rates

In Fig. 3.4 we illustrate the total number of legitimate and attack DNS requests reaching the Authoritative DNS Server for various sampling rates at the SDN Switch. We began feeding the Attack Traces at 30 sec. We observe that our schema mitigates most of the attack traffic in 60 sec regardless of the sampling rate at the SDN Switch. Benign IP addresses were not misclassified due to the zero FN's of BF's. The ADC, described in Section 3.4, included (i) a BF of 9.98 KB (1% FP's), (ii) a SymSpell dictionary of 4.05 MB and (iii) 2 CMS's with 5 hash functions and 10,000 counters of 16 bits per hash function hence 97.66 KB per sketch (0.02% estimation

error with 97% certainty). Thus, a total of 4.25 MB were used to protect a moderate-size zone of 8,303 RR's.

3.5.2.3 DNS Firewall (DFW)

In the following we examine whether the deployment and lookup time of BF's and thus, our DFW's, are suitable for the stringent latency requirements of cloud services. BF's (probabilistic data structures) are compared against Red-Black Trees (RBT's) that are deterministic data structures used by BIND to represent the contents of the Authoritative DNS Server zones in memory [119]. Our DFW achieved 25 Kpps throughput. This may seem a small rate, but higher performance may be achieved via load balanced Docker Containers (Fig. 3.3).

In our experiments, the DFW was fed with an indicative total rate of 25 Kpps, with 12.5 Kpps corresponding to the Legitimate Traces and 12.5 Kpps to the Attack Traces. Using either a BF or an RBT as the filtering mechanism for various zone sizes, we tested how fast the data structure is filled with all the valid names (including "ntua.gr" zone names) and thus, is able to filter invalid requests. Moreover, we examined how the zone size affects the DFW capability to forward all requests. Recall that our BF utilized 5 non-cryptographic hash functions. DFW was deployed on a Virtual Machine (VM) with 2 CPU cores and 4 GB RAM. The physical machine is a Dell PE R730 with Intel Xeon E5-2620 v3 2.4GHz.

Table 3.2 exhibits deployment time and average number of valid requests forwarded by the DFW. We observe that BF deployment times are lower than those of RBT for increasing zone sizes. Moreover, BF's forward all valid requests, i.e. 12.5 Kpps, regardless of the zone size. Conversely, RBT's containing bigger zones forward less valid requests. Insertion and lookup time in BF's is proportional to the number of used hash functions, whilst in RBT's logarithmic with respect to zone size. Thus, BF's outperform RBT's for larger DNS zones.

Number of FQDN Entries	**Deployment Time (sec)**		**Average Number of Forwarded Valid Requests (Kpps)**	
	BF	**RBT**	**BF**	**RBT**
8,303	0.033	0.047	12.5	12.49
1,470,834	9.955	25.367	12.5	10.55
4,905,628	31.389	81.695	12.5	10.26

Table 3.2: Deployment time and average number of forwarded valid requests - Comparison between Bloom Filters (BF's) and Red-Black Trees (RBT's)

3.6 Summary & Concluding Remarks

We proposed a privacy-preserving collaborative schema to protect Authoritative DNS Servers from Water Torture attacks tailored to the requirements of cloud services. Our approach employed space-time efficient data structures and algorithms in the user space to detect and mitigate such attacks. Specifically BF's were primarily used to map DNS zone RR's taking advantage of their space, time and privacy properties. Our proof of concept setup and evaluation experiments were based on traces generated from publicly available DNS traffic datasets. Our results

indicated that the proposed schema may efficiently secure Authoritative DNS Servers without requiring extensive resources.

This chapter mainly focused on demonstrating that BF's are suitable for efficiently mapping Authoritative DNS Server zone names and effectively filtering DNS requests pertaining to Water Torture attacks. However, user-space methods were employed; these may be unable to cope with the increased rate of modern DDoS attacks. In Chapter 4, we will adapt our proposed BF-based mitigation mechanism to higher attack rates by utilizing data plane programmability techniques and specifically the eXpress Data Path (XDP) framework. Therefore, our schema will be capable of efficiently differentiating between legitimate and malicious DNS requests within the data plane of Authoritative DNS Servers.

Chapter 4

XDP-based Acceleration of Bloom Filter Lookups for Efficient Data Plane Mitigation of DNS Attacks

In this chapter[1], we utilize eXpress Data Path (XDP) for DNS Deep Packet Inspection (DPI) in order to mitigate Water Torture attacks at the NIC driver level of Authoritative DNS Servers. Our approach may benefit DNS administrators who wish to filter attack traffic within their DNS infrastructure and avoid the latency overhead and additional costs imposed by external cloud scrubbing services. Our schema does not depend on specialized hardware and does not blacklist entire domain name suffixes, hence does not block legitimate requests.

In our proposed schema, packets are intercepted by XDP that identifies messages of DNS requests for further processing. Requested names are extracted from the message payload and categorized based on their validity. Valid names are forwarded to the user space to be resolved, whilst invalid ones are dropped within the Linux kernel at an early stage without downgrading the DNS service. Names are classified using BF's that map DNS zone contents in a memory-efficient manner. Recall that these probabilistic data structures are free of False Negatives (FN's) and therefore valid DNS requests are never dropped.

We provide a proof of concept setup to test our schema under a DDoS attack scenario and assess how mitigation performance is affected by DPI on DNS requests. Our experiments verify that using XDP significantly increases the throughput of valid DNS responses compared to user space alternatives. In conclusion, XDP emerges as a promising solution for the mitigation of Water Torture attacks against DNS servers.

4.1 Motivation

Recent terabit-scale DDoS attacks [120] highlighted the shortcomings of traditional mitigation policies. To remedy these limitations, as mentioned in Chapter 2, promising data plane programmability methods have been proposed. Notably, eXpress Data Path (XDP) [10] was sug-

[1]This work is included in the proceedings of the 6th IEEE International Conference on Network Softwarization (IEEE NetSoft 2020)

gested to implement high throughput protection mechanisms, e.g. Cloudflare's DDoS mitigation system [121,122]. XDP provides a programmable network data path within the Linux kernel that handles ingress packets at the NIC driver level via an extended Berkeley Packet Filter (eBPF) program.

As previously mentioned, one of the most common DDoS attack targets concerns the DNS infrastructure [2]. Specifically, Water Torture [4] attacks to Authoritative DNS Servers are singled out in terms of sophistication and devastating impact. Recall from Chapter 3 that this attack vector utilizes invalid Fully Qualified Domain Names (FQDN's) to exhaust the Authoritative DNS Server processing capacity. The attacker exploits effectively the available botnet resources by issuing requests that are generated at random. This ensures that attack traffic is not cached in intermediary Recursive DNS Servers (Recursors) and thus, all requests are delivered to the victim Authoritative DNS Server. These attacks are typically mitigated via external cloud scrubbing infrastructures, e.g. Akamai [123]. However, such facilities introduce additional costs and are usually located far from the victim premises. Therefore, latency of ingress packets may significantly increase.

We propose a schema that relies on XDP to efficiently mitigate Water Torture attacks within Authoritative DNS Servers. Thus, administrators may avoid external scrubbing services in favor of in-house solutions. Our approach consists mainly of an eBPF program that distinguishes valid and invalid DNS requests entirely within the Linux kernel at the NIC driver level, while maintaining high-throughput packet filtering.

We intercept ingress traffic and process it via an eBPF program on a per-packet basis inspecting the payload of DNS requests. Such a DPI mechanism isolates DNS requests and extracts the corresponding FQDN from the question section in the payload. Rapid name lookups are accomplished via BF's that map DNS zones in the Linux kernel in a space-efficient manner. Requests pertaining to FQDN's not present in the DNS zones are dropped, whereas valid ones are passed to the user space to be resolved. Our implementation is available from [124].

The remainder of this chapter is structured as follows: Section 4.2 describes related work; Section 4.3 provides a high-level overview of our mechanism; Section 4.4 discusses implementation details; Section 4.5 includes the evaluation of our schema. Finally, Section 4.6 summarizes our work.

4.2 Related Work & Contributions

In the previous chapter (see Chapter 3) we relied on BF's to handle DNS requests from Water Torture attack sources in cloud infrastructures. BF's were used to map the names of large DNS zones and filter suspicious DNS traffic accurately and efficiently, e.g. 5 million names required BF's of about 17 MB. In this chapter, we extend our BF-based mechanism by using XDP to improve our filtering component performance and hence provide efficient mitigation within the Authoritative DNS Server. Mitigation actions are performed entirely at the NIC driver level within the Linux kernel contrary to related works on Water Torture that do not employ data

plane programming [125, 126] or require specialized hardware [53]. Our schema provides in-house mitigation instead of outsourcing it to external cloud scrubbing services that may increase packet latency. Moreover, we apply XDP for DNS DDoS attack mitigation as DNS is a fitting use case to test XDP DPI capabilities that are not studied in other works on XDP [127, 128].

Our work is motivated by the wide adoption of Programmable Data Planes (PDP's) that achieve significant packet throughput improvements for various networking applications [10, 127, 129, 130]. Notably, Cloudflare [121] relies on XDP to define mitigation rules in the Linux kernel and filter offensive traffic. Their bpftools toolset includes a utility [122] that drops all DNS requests pertaining to a specific domain name suffix when excessive NXDOMAIN responses are detected. Moreover, Facebook introduced Katran [131], an XDP-based load balancer that efficiently distributes received traffic across the available service instances.

XDP and eBPF have been used by Cilium [132] to enforce security and network policies for deployed containers. Moreover, the open-source IDS Suricata [133] includes XDP-based plugins that are used to drop packets matching a predefined pattern. Finally, in [134] a prototype is presented that complements eBPF with a Lua-based XDP schema; the user-space DNS software is substituted by in-kernel name resolution that does not implement filtering of DNS DDoS attacks.

4.3 Proposed Filtering Mechanism: High-Level Overview & Design Features

This section provides a high-level overview of our proposed schema and lists its main design principles.

4.3.1 High-Level Description

Fig. 4.1 depicts a high-level description of our schema. Incoming traffic is intercepted by an XDP Hook that is attached on the Authoritative DNS Server NIC driver level. Received packets are processed by an eBPF program [135] that we customized for the DNS Water Torture attack mitigation use case. This isolates DNS requests and forwards the remaining traffic, i.e. DNS responses and/or non-DNS traffic, to the User Space through the Socket Buffer and the Network Stack. The question section of DNS requests is parsed to extract the FQDN that is hashed multiple times (typically 3-10).

These hashes are used to determine if this name is valid, i.e. present in the Authoritative DNS Server zones or not. DNS zones are contained in an eBPF map in the form of a BF. This map is initialized by the user-space program, i.e. the XDP Control Plane, when the eBPF program is attached in the Linux kernel. Invalid DNS requests are dropped, whilst valid ones are forwarded to the User Space through the Network Stack to be resolved by the DNS Server Software.

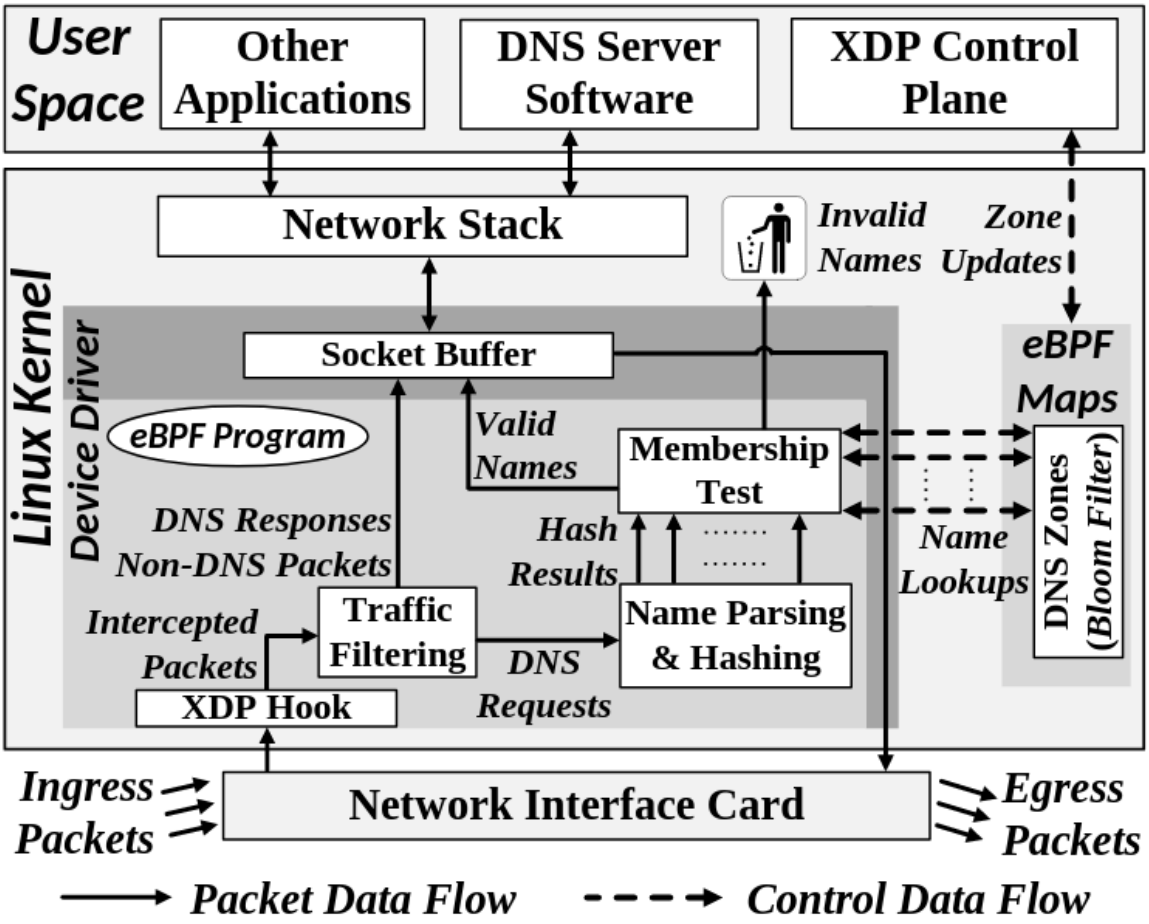

Figure 4.1: Baseline design of the proposed mitigation approach

4.3.2 Design Features

The main design features of our approach are the following:

- **In-network, high-throughput packet filtering of received DNS requests:** We use XDP to differentiate between valid and invalid FQDN's entirely at the NIC driver level. Invalid requests are detected and dropped at the earliest possible layer in the Linux kernel, i.e. before the system allocates any memory to them. This alleviates the DNS software that operates in the user space from the burden of processing large amounts of useless data.
- **Independence from specialized hardware and DNS Software:** Our XDP-based schema may adapt to various generic infrastructures regardless of the DNS software, e.g. BIND, PowerDNS, used for resolving incoming DNS requests. Our approach does not require specialized hardware contrary to P4 [12] and DPDK. Native XDP depends on the NIC driver version and thus, only a compliant driver update may be required.
- **Efficient DNS request filtering with no generic blacklisting:** We use BF's to effectively map zone FQDN's in the Linux kernel. BF's enable memory-efficient rapid element lookups, consistent with the XDP low latency packet processing. The zero FN property of BF's ensures that all valid requests are delivered to the user space to be resolved. Moreover, a tolerable FP probability guarantees that only a small, regulated fraction of invalid

requests are forwarded. These properties allow a fine-grained classification of requested names contrary to approaches that blacklist entire domain suffixes [122, 125] or Recursors that may forward attack traffic.

- **Cost-effective hashing in the system kernel:** Our approach leverages on efficient hash functions to hash names available from the question section of the received DNS requests. We selected the MurmurHash3 (Mmh3) algorithm as non-cryptographic hash funcions are evaluated faster than cryptographic ones. Instead of performing separate hash calculations, we reduce the overall computation cost by enabling Double Hashing [91] (see subsection 2.6.1.1).

4.4 Implementation Details

This section includes implementation details about our schema. We elaborate on: (i) the XDP Data Plane Program (XDP DPP) that enables packet filtering in the Linux kernel (subsection 4.4.1) and (ii) the XDP Control Plane Program, the user-space utility that manages the XDP DPP (subsection 4.4.2).

4.4.1 The XDP Data Plane Program (XDP DPP)

We implemented the XDP DPP in a restricted version of C. In the following, we provide more details pertaining to ingress traffic filtering (subsection 4.4.1.1), name parsing and hashing (subsection 4.4.1.2) as well as name membership lookups based on BF's (subsection 4.4.1.3).

4.4.1.1 Ingress Traffic Filtering

This stage involves inspecting if an intercepted packet includes a DNS request or not since the Authoritative DNS Server may receive non-DNS packets and/or DNS responses to issued requests. Thus, layer 2-4 headers are parsed to identify UDP/TCP packets directed to destination port 53. Parsing the header of a specific protocol layer requires checking that no data are read out of bounds [10]. This requirement is imposed by the eBPF verifier during compilation to guarantee system safety.

Received packets not corresponding to DNS requests are forwarded to the user space through the kernel network stack without further processing. Regarding DNS requests, the eBPF program locates the DNS payload beginning and hence the question section.

4.4.1.2 Name Parsing & Hashing

This stage involves extracting the requested FQDN from the question section of the received DNS message and hashing it several times via Mmh3 to provide the inputs required by the BF lookups. The purpose behind selecting Mmh3 is twofold:

- It is a non-cryptographic hash function and hence requires fewer arithmetic operations than cryptographic ones, e.g. MD5, etc. This is important as the eBPF verifier restricts the XDP

DPP total size to guarantee the kernel safety. Hash functions require fewer computations, extending the capabilities of our program.

- It supports calculation of several independent name digests by varying the provided seed. Therefore, we avoid using more than one hash algorithms. Our implementation was based on code available in GitHub [136].

XDP DPP mainly involves a bounded loop that parses and hashes names. This loop is unrolled during the program compilation up to the maximum supported FQDN length, i.e. the loop is replaced by the program instructions corresponding to each iteration. This method enables the eBPF verifier to guarantee that the loop terminates and authorize its execution, whilst the loop overhead is significantly reduced. However, the number of preallocated eBPF program instructions increases.

The loop in our program iterates up to the maximum supported FQDN length. In each iteration, the necessary checks are performed to ensure that no data are read out of bounds. The ending of parsed FQDN's is denoted by zero. The parsed characters are added to chunks and h_1, h_2 values required by the Double Hashing technique (see subsection 2.6.1.1) are updated every 4 loop iterations because Mmh3 hashes strings in chunks of 4 Bytes [136]. When the loop terminates, the trailing characters are hashed to calculate the final digests d_i using Double Hashing.

4.4.1.3 Name Membership Lookups based on Bloom Filters

The final stage checks if the received FQDN is included in the DNS zones. Recall that we use BF's to map the entire Authoritative DNS Server zone contents. These are represented as eBPF maps within the Linux kernel [128]. We opted for maps of type BPF_MAP_TYPE_ARRAY that define arrays of constant size optimized for fast lookups.

A limitation of the C programming language is that arrays of bits, used by BF's, are not directly supported. Thus, 1-bit sized variables were encapsulated within 8-bit character types [128]. When the desired bit is located in the BF, appropriate right shifting operations place it in the least significant bit position.

4.4.2 The XDP Control Plane Program

We implemented a C program that acts as the XDP Control Plane for our eBPF program and is mainly responsible for: (i) invoking the eBPF verifier that ensures the normal operation of the system before the XDP program is loaded, (ii) attaching the eBPF program as an XDP Hook at the Linux kernel NIC driver level and (iii) inserting names, mapped to BF's, in eBPF maps.

4.5 Evaluation

In this section we concentrate on XDP DPP performance under a Water Torture attack scenario on an Authoritative DNS Server. Specifically, we investigate how our system is affected by DPI

actions on ingress DNS requests. This trait differentiates us from [127, 128] that rely on layer 2-4 headers. We first assess design features of our eBPF program and then report experimental results.

4.5.1 Assessment of Critical Design Features

Developing a prototype implementation of the XDP DPP (Fig. 4.1) requires assessing critical design features. We first present the rationale of selecting BF sizes and hash function number. We then study the capability of our schema to handle FQDN's of various lengths to assess XDP appropriateness for DNS DPI.

4.5.1.1 Bloom Filter Sizes for In-Network Membership Lookup

XDP DPP performance is mainly affected by the number of hash function calculations used for BF membership lookups. We selected this critical design parameter by considering the appropriate range of hash functions based on BF memory and accuracy specifications.

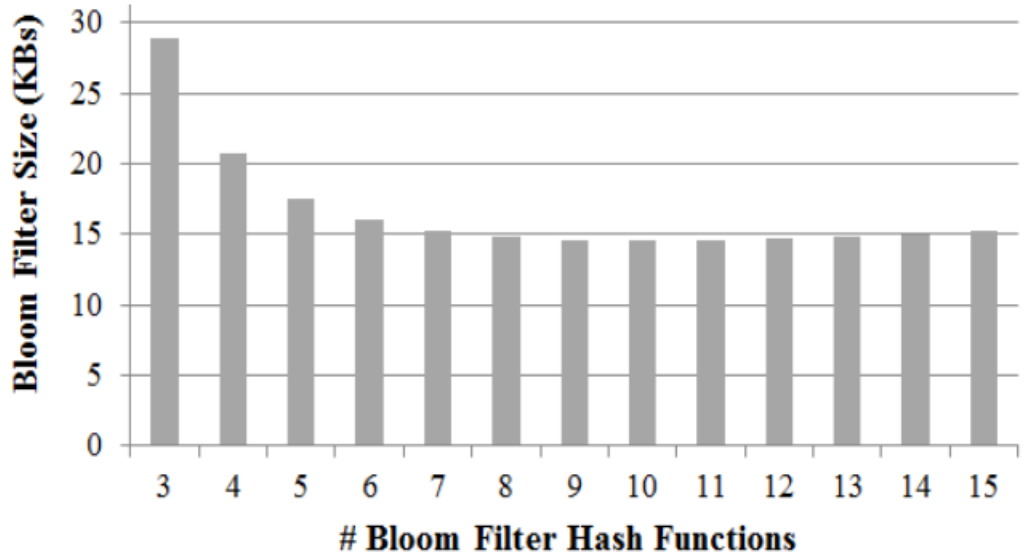

Figure 4.2: Required BF sizes varying the number of hash functions used

Fig. 4.2 depicts the required BF sizes for a targeted FP percentage of 0.1% varying hash function number between 3 and 15. Our BF's contain 8,303 names obtained via AXFR requests within our campus network zone "ntua.gr".

We observe that BF's may accurately map the "ntua.gr" zone employing at most 28.66 KB when only 3 hash functions are utilized, whilst BF size levels at 14.57 KB when 10 hash functions are used. Thus, we will focus on evaluating our schema using between 3 and 10 hash functions; this range is also applicable to bigger DNS zones (see Chapter 3 and [137]). Usage of more hash functions may lead to oversized BF's, increasing lookup latency without dropping FP percentages.

4.5.1.2 Maximum Supported FQDN Length

In the following we determine the maximum FQDN lengths that our schema is capable of parsing. Recall that the size of our program is restricted by the eBPF verifier that guarantees the safety of the kernel. The XDP DPP size is mainly affected by the total number of hashing operations. As their number increases, the maximum supported FQDN length decreases, thus restricting the scalability of our schema.

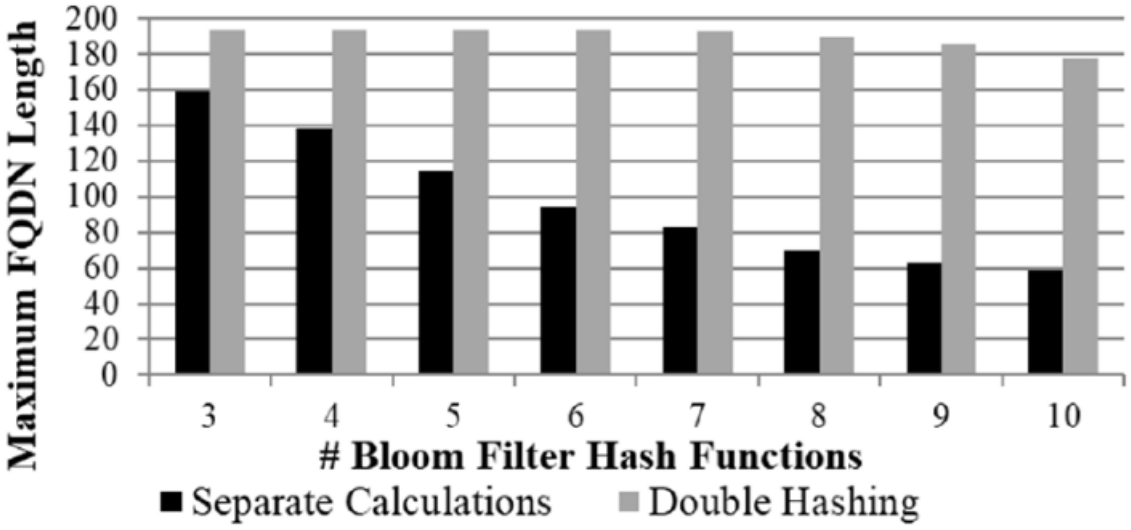

Figure 4.3: Maximum supported FQDN length determined by the number of used hash functions employing either Double Hashing or separate calculations

Fig. 4.3 depicts the maximum FQDN length allowed by the eBPF verifier when hash functions range between 3 and 10 (subsection 4.5.1.1). We considered (i) separate calculations of all required hash functions and (ii) usage of Double Hashing to speed up the process.

We observe that the eBPF verifier does not impose remarkable limitations on the permitted FQDN sizes and thus, XDP may support a wide variety of DNS zones. Specifically, separate hashes limit FQDN lengths to 159 characters when 3 hash functions are used and 59 characters for 10 hash functions. This is significantly improved by using Double Hashing which supports 194 characters when employing 3 hash functions that slightly drops to 178 characters for 10. Recall that Double Hashing includes less calculations (see subsection 2.6.1.1) and thus, preallocates fewer program instructions in the compiled eBPF program.

4.5.2 Throughput of the Proposed Mitigation Schema

This subsection describes our experimental setup, details the traces used as inputs and reports on our schema performance.

4.5.2.1 Testbed Overview

Our laboratory testbed included 2 VM's: The Traffic Generator and the Authoritative DNS Server. The former used Tcpreplay to forward traces to the Authoritative DNS Server. The latter

implemented the modules of Fig. 4.1; the NIC Device Driver executed the XDP DPP (eBPF program) in native mode, fed by the XDP Hook. Both VM's included the Debian 10 Operating System (OS) with the Linux kernel version 4.19 and shared a standard Gigabit interface. The Authoritative DNS Server comprised of 1 CPU core, 2 GB RAM, 2 NIC queues and Virtio adapters. The physical machine was a Dell PE R730 with Intel Xeon E5-2620 v3 2.4 GHz. Finally, we selected BIND 9 as the DNS Server Software as it is one of the most frequent options in DNS.

4.5.2.2 Legitimate & Attack Traces

DNS request filtering is based on the validity of related names, whilst the FQDN length is the major factor affecting mitigation throughput; this defines the total hash function calculations required. We created custom traces involving FQDN's with lengths matching our campus zone ("ntua.gr") for normal traffic and [53] for attack traffic.

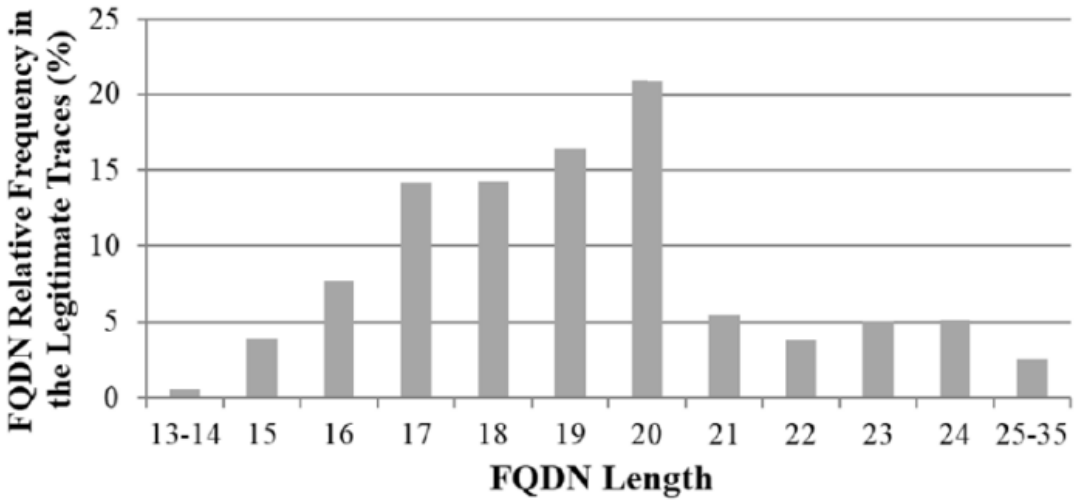

Figure 4.4: Distribution of FQDN length (in characters) within the Legitimate Traces

Legitimate Traces involved DNS requests pertaining to names randomly sampled from our "ntua.gr" zone. To that end, we gathered 8,303 names via AXFR requests. The length of the collected FQDN's varied between 13 and 35 characters. As expected, their length was not uniformly distributed. Fig. 4.4 depicts the distribution of FQDN lengths in the constructed dataset.

We constructed Attack Traces based on the FQDN lengths tested in [53]. Traces consisted of DNS requests involving names with randomly generated FQDN prefixes, whilst a name is never repeated. We randomly sampled DNS requests with FQDN lengths uniformly distributed between 20 and 32 characters; the variable name prefix may comprise of characters from the Latin alphabet, decimal digits and/or hyphens.

4.5.2.3 Valid DNS Response Throughput Measurements

Legitimate and Attack Traces were fed to the Authoritative DNS Server in parallel at rates of 2 and 125 Kpps respectively. The Legitimate Traces rate was based on SWITCH Authoritative

DNS Server data [118]; each Authoritative DNS Server resolves about 2 Kpps of requests. The Attack Traces rate was selected to be within the processing capabilities of our testbed 1 Gigabit NIC and able to severely downgrade the DNS service. Overall, these traces drove our schema to a throughput ranging to 50% of an XDP Program that simply forwards packets to the user space through the Linux kernel without performing DPI.

We varied hash functions between 3 and 10 (subsection 4.5.1.1) and BF accuracy was set to 0.1% FP's. Fig. 4.5 depicts the ratio of valid DNS responses forwarded by the Authoritative DNS Server after mitigation employing either (i) the XDP DPP with Double Hashing, or (ii) the XDP DPP using separate calculations, or (iii) our previous work Mmh3-based user-space utility [137] (described in Chapter 3). We also depict the valid DNS response throughput when mitigation is not applied.

Mitigation based on the XDP DPP using Double Hashing outperforms the one with separate calculations as the hash function number increases. The former approach is capable of achieving a satisfactory trade-off between the required BF size and valid DNS response throughput. Specifically, it safeguards the normal operation of the Authoritative DNS Server by forwarding between 94.36% (10 hash functions) and 97.02% (3 hash functions) of valid DNS responses. Moreover, the XDP DPP using Double Hashing clearly exceeds our user-space utility [137] capabilities that forwards approximately the 13.5% of legitimate responses. Without mitigation, the Authoritative DNS Server forwards only the 7.34% of valid DNS responses.

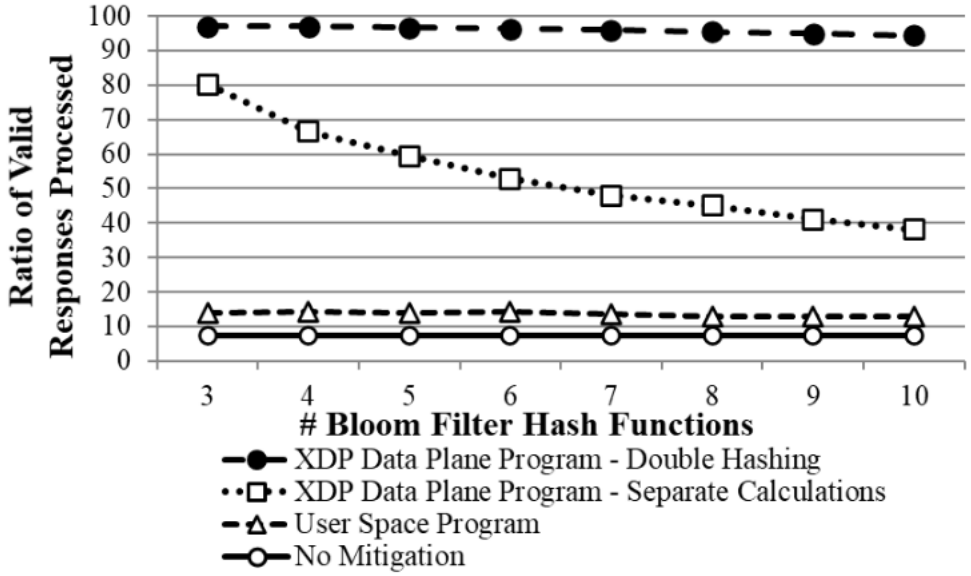

Figure 4.5: Valid DNS response throughput as a function of BF hash functions

4.6 Summary & Concluding Remarks

We leveraged on the XDP DPI capabilities to enable high-throughput Water Torture mitigation within Authoritative DNS Servers. We implemented a prototype that isolated DNS requests; requested names were extracted from the message payload and subsequently hashed for membership lookups. To that end, we utilized BF's in the Linux kernel to map DNS zones efficiently

in terms of space and time. DNS requests for names not included in the BF were dropped early at the Authoritative DNS Server NIC driver, whilst the remaining ones were resolved in the user space.

Our experiments verified the trade-off between the BF size and the hash function number that ultimately affect the XDP DPP throughput. As hash functions increase to a certain number [91] the BF size is reduced, but the mitigation throughput deteriorates. Note that, our approach was tested on a VM with a single CPU core to provide results for the simplest possible setup. DNS administrators may use more cores to enhance filtering throughput [10], e.g. our schema may sustain attack rates up to 200 Kpps using 2 cores.

This chapter focused on accelerating BF lookups within the data plane of Authoritative DNS Servers for efficient mitigation of DNS DDoS attacks. Our developed data plane mechanism significantly improved the filtering throughput of our user-space schema presented in Chapter 3. Although attack mitigation was accomplished efficiently on-premises, DDoS attacks are usually mitigated more effectively closer to their sources, e.g. within Recursors (for our considered DNS DDoS attacks) or upstream scrubbing services. However, our BF-based mitigation schema cannot be applied to external filtering appliances unless the contents of Authoritative DNS Server zones are available; such information is usually restricted for security reasons because zone transfers may disclose sensitive data about the users and the services of a network. In the following chapter, we will present a privacy-aware mechanism for distributing Authoritative DNS Server zone contents.

Chapter 5

Enabling Privacy-aware Exchanges for Authoritative DNS Server Zones

In this chapter[1], we propose a privacy-aware schema that enables Authoritative DNS Servers to distribute their zones to third parties, e.g. Recursive DNS Servers (Recursors) or upstream scrubbing services, without disclosing sensitive information. Therefore, DNS DDoS attack mitigation may be effectively accomplished at external vantage points, presumably closer to the attack sources than the Authoritative DNS Server. Moreover, DNS administrators may construct inclusive name whitelists to accurately filter malicious traffic associated with botnet activities, e.g. DGA-related DNS messages. Our schema leverages on the space, time and privacy-enhancing properties of Cuckoo Filters (CF's) to map zone names in an efficient manner, while permitting rapid name updates for large zones.

The feasibility of our approach is tested via experiments within our laboratory testbed for a variety of DNS zones. Our evaluation intends to assess the privacy-awareness of our schema and its responsiveness to zone name changes. We conclude that our approach enables mapping of large DNS zones, while preserving privacy.

5.1 Motivation

Recursors are commonly abused by DNS attacks as intermediaries to victim Authoritative DNS Servers [28]. In particular, as mentioned in the previous chapters, Water Torture attacks [4] aim at exhausting computational resources of Authoritative DNS Servers by forwarding an immense number of malicious DNS requests through Recursors. Typically, these involve randomly generated Fully Qualified Domain Names (FQDN's), appropriately crafted to be requested once and thus, bypass the DNS caches of Recursors. As a collateral damage, services offered by Recursors are degraded since response latency and consumed memory resources increase [53]. Apart from DDoS attacks, Recursors may also be abused by DGA's to establish botnet communications between infected devices (bots) and their C&C servers.

Flooding attacks can be mitigated more efficiently close to their origins [33]. However,

[1] This work is included in the proceedings of the 2020 ACM/IRTF Applied Networking Research Workshop (ANRW 2020)

a complete list of zone FQDN's is commonly not available to Recursors as full zone transfers, i.e. AXFR type requests are typically restricted by Authoritative DNS Servers for security reasons [8, 138]. Thus, effective filtering policies near the attack sources cannot be enforced. Furthermore, DNS administrators wishing to filter DGA traffic within their networks are unable to build accurate name whitelists. Thus, they typically depend on anomaly-based methods, e.g. Machine Learning (ML) algorithms, which may result in significant amounts of legitimate traffic being dropped.

In this chapter we propose a privacy-aware schema to facilitate the efficient distribution of Authoritative DNS Server zones to third-party filtering appliances, e.g. Recursors and/or upstream scrubbing services [139], while being compatible with the existing DNS infrastructure. Our mechanism may benefit Recursors that wish to mitigate DNS flooding attacks within their premises, thus safeguarding the Quality of Service (QoS) offered to their end-users. Moreover, our approach may enable DNS administrators to filter botnet traffic, e.g. DNS messages generated by DGA's, more accurately by collecting zone contents from multiple Authoritative DNS Servers and deploying effective security mechanisms.

We leverage on Cuckoo Filters (CF's) [19] to map valid zone names in a hashed format, thus FQDN's are not exposed as plaintext. CF's are time and space efficient probabilistic data structures that enable rapid element lookups in storage and bandwidth constrained applications. False Positives (FP's) are possible, resulting in the Authoritative DNS Server forwarding a regulated volume of NXDOMAIN responses, whilst False Negatives (FN's) cannot occur for appropriately configured CF's, hence valid DNS requests are never dropped. Contrary to Bloom Filters (BF's) [16] that require rebuilding the entire data structure, CF's support dynamic item deletions. Therefore, they are suitable candidates for DNS attack mitigation services that require frequent updates and cannot tolerate any downtime. Our implementation is available from [140].

The remainder of the chapter is structured as follows: Section 5.2 presents related work; Section 5.3 provides an overview of our schema; Section 5.4 involves implementation details; Section 5.5 includes the evaluation of our mechanism; Section 5.6 summarizes our work.

5.2 Related Work & Contributions

Various approaches reported in the literature suggest probabilistic data structures (BF's) to enhance DNS performance and security. An Internet Engineering Task Force (IETF) draft [141] proposed using BF's to map DNSSEC zone names in a space-efficient, privacy-preserving format for accelerating authenticated responses to requests about invalid FQDN's. However, this proposal may require tools external to DNS [142], i.e. a separate web server that contains BF's, and may not directly support incremental updates as item deletions are not possible in standard BF's.

In [143], it is recommended that Recursors use BF's for monitoring DNS requests, thus relying on privacy-aware summaries of sensitive DNS data. The space-efficient and hashed nature of BF's enables logging information over long time periods, whilst end-user privacy is not

compromised. Thus, General Data Protection Regulation (GDPR) [99] directives are not violated. Recursor administrators are required to issue targeted requests over their collected BF's to determine if newly blacklisted domains have occurred within their network, triggering related actions.

Hosts controlled by a particular botnet are based on the same Domain Generation Algorithm (DGA) [6] to contact their Command and Control (C&C) servers. Thus, they typically request the same invalid FQDN's. Based on this, [144] maps invalid names requested within a network to BF's on a per host basis. BF bits from diverse hosts are compared to detect similar behavior and subsequently identify abused hosts without logging any personal data. In addition, BF's are used to map valid names and efficiently locate C&C servers.

Malicious activities related to DNS are usually based on randomly generated names via DGA's. To remedy them in operational environments, some Recursor software implementations consider incorporating BF functionality. Notably, PowerDNS [145] uses BF's to track newly observed names and detect those related to DGA's. FQDN's appearing for the first time are further inspected, whilst reappearing names are considered for resolution. Similarly, Unbound [146] includes a learning mode that collects valid FQDN's from NOERROR responses and stores them in BF's. When a predefined threshold is reached, the filtering mode is activated and unknown names are dropped. Our proposed schema could significantly increase the accuracy of these approaches by distributing valid Authoritative DNS Server zone names, while preserving privacy.

Approaches related to DNS Water Torture mitigation employ either ML [5,53,126] or sketch-based methods [125, 137]. In previous work (Chapter 3), we relied on BF's to mitigate Water Torture attacks and outlined their privacy-preserving properties for outsourcing Authoritative DNS Server zone protection to third parties, e.g. Recursors and cloud scrubbing services. In this chapter, we extend the schema presented in Chapter 3 by implementing a privacy-aware mechanism that enables Authoritative DNS Servers to communicate their zones in a hashed format, thus distributing mitigation services. Contrary to Chapter 3 and related approaches [141, 143,144] we use CF's, instead of BF's, to reduce our privacy-aware DNS zone sizes and support incremental updates.

5.3 Proposed Approach: Design Features & Baseline Design

This section presents an overview of our mechanism. The main design features of our schema are outlined in subsection 5.3.1, whereas the baseline design of our approach is described in subsection 5.3.2.

5.3.1 Design Requirements

The main design requirements of our schema are:

- **Privacy-aware distribution of Authoritative DNS Server zone FQDN's**: The desired

system should map the Authoritative DNS Server zone names not in their actual form, but hashed. This will enable Recursors, upstream scrubbing services or other filtering appliances to retrieve a complete list of all valid FQDN's without inferring the exact zone contents. Thus, fine-grained filtering policies may be enforced closer to the DNS attack origins (e.g. bots external to the Authoritative DNS Server network).

- **Efficient Authoritative DNS Server zone mapping**: Our schema should exhibit: (i) compact storage of hashed FQDN's within Authoritative DNS Servers, (ii) low latency filtering of malicious DNS requests and (iii) bandwidth conservation during information transmission among Authoritative DNS Servers and Recursors or other filtering appliances.
- **Compatibility with the existing DNS infrastructure**: We require a suitable data serialization format to map hashed FQDN's within Authoritative DNS Servers with minor software modifications. Copies of the privacy-aware zones may be obtained using widely-adopted types of DNS requests, e.g. AXFR [30] and IXFR [31].
- **Support for incremental updates**: DNS attack mitigation systems should be able to update their filtering policies incrementally without downloading again the entire information included in the privacy-aware DNS zones. Thus, the desired system should support flexible name updates.

5.3.2 High-Level Description

Complying with the aforementioned requirements, Fig. 5.1 provides a baseline description of our schema. Plaintext DNS Zones (PltZn's) contain the Resource Records (RR's) that are under the Authoritative DNS Server management responsibility. These zones may receive manual and/or Dynamic DNS updates [147] either by the Authoritative DNS Server administrator or Subscribed Devices, e.g. a Dynamic Host Configuration Protocol (DHCP) Server. Details related to the modified RR's are subsequently recorded in the Zone Updates Log.

Our system relies on the Privacy-Aware Zone Manager (PAZM), responsible for constructing and maintaining the privacy-preserving versions of the PltZn's. The PAZM recovers a list of the entire plaintext RR's and hashes their corresponding FQDN's to create the Hashed DNS Zones (HsZn's). Moreover, recently modified RR's along with details pertaining to them are retrieved from the Zone Updates Log. Sensitive information (i.e. names) is hashed, enriched with metadata and included in an Incremental DNS Zone (IncZn) that reflects recent zone changes. The contents of the HsZn's are renewed in frequent time intervals to match PltZn current contents.

Recursors (or DNS administrators in general) that wish to filter malicious DNS requests within their infrastructure may retrieve the necessary zone names in a privacy-aware format from the Authoritative DNS Server. This is possible by initially getting a full copy of an HsZn along with its recent modifications from the corresponding IncZn. At regular time intervals, DNS administrators may use the IncZn contents to incrementally update their Filtering Modules. Thus, they avoid the time consuming process of collecting again the entire contents of HsZn's.

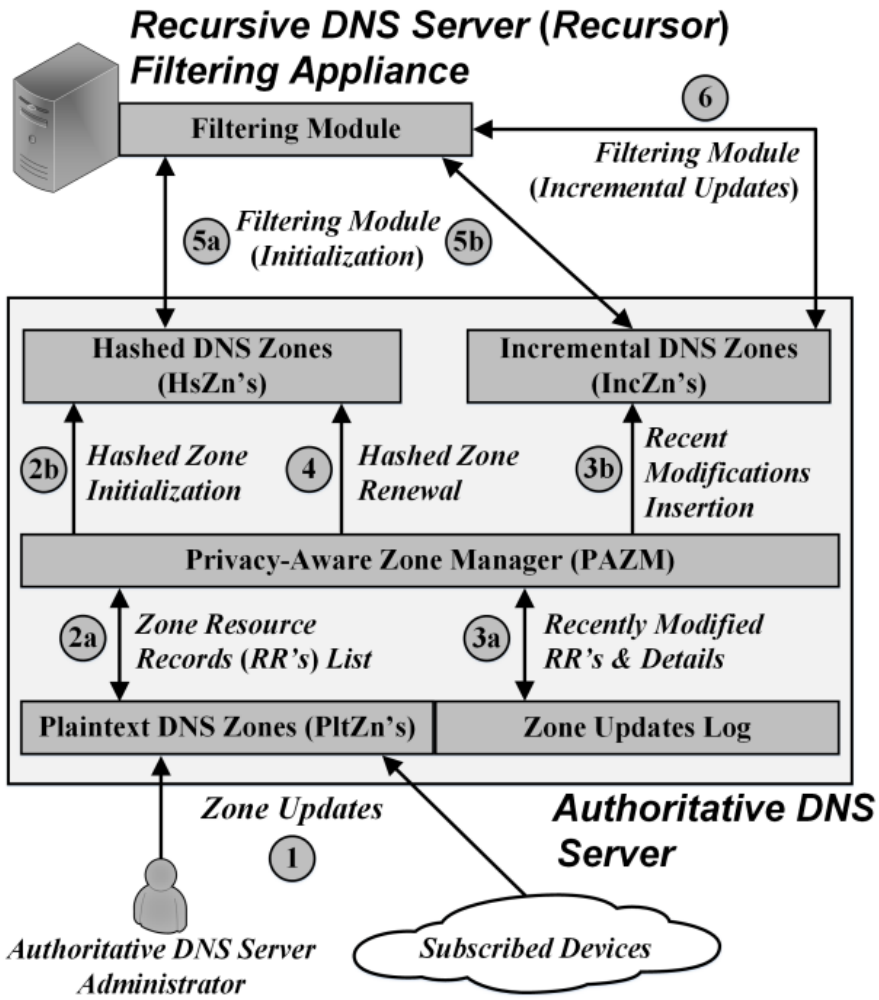

Figure 5.1: Baseline design of our proposed schema

5.4 Implementation Details

This section involves implementation details regarding our schema. Specifically, we describe the PAZM module and the properties of zones introduced in our architecture, i.e. Hashed DNS Zones (HsZn's) and Incremental DNS Zones (IncZn's).

5.4.1 Privacy-Aware Zone Manager (PAZM)

This module is responsible for building and maintaining the CF's whose fingerprints are used to create and revise the privacy-aware DNS zones. The PAZM is implemented in Python and relies on a CF implementation [148] that we customized to manipulate CF contents as desired. CF's in [148] employ the MurmurHash3 (Mmh3) algorithm [149] for both $fgp()$ and $hash()$ (see CF background of subsection 2.6.1.2).

Initially, the PAZM retrieves the Plaintext DNS Zone (PltZn) RR's and retains their FQDN. These are hashed into fingerprints to fill the entries of a CF, thus creating the HsZn. Notably, an FQDN may appear multiple times in the zone, hence only its first appearance is hashed and

inserted in the CF.

After HsZn is initialized, the PAZM frequently recovers recent PltZn changes from the Zone Updates Log. The corresponding names are used to update CF contents and changes are incorporated in the IncZn via Dynamic DNS updates. The PAZM ignores changes pertaining to: (i) RR's whose value was updated, thus their FQDN was not modified and (ii) RR's that share an FQDN with others, but differ in RR type and/or value. The latter case is handled by associating frequency counters with each distinct FQDN. These monitor how many times an FQDN is inserted in the PltZn and thus, determine names newly inserted or completely removed from the zone. Otherwise, an FQDN could have been inserted multiple times in the CF resulting into significant memory overhead.

5.4.2 Hashed DNS Zones (HsZn's)

These zones hold the FQDN's of the PltZn's hashed and mapped in CF's. Recursors may retrieve full copies of HsZn's by performing AXFR type DNS requests. Listing 1 depicts the HsZn information serialization format. CF-related information is encapsulated by the PAZM within RR's of TXT type. Note that generic DNS parameters that are not specific to our proposed mechanism, e.g. TTL value and standard zone RR's (SOA, NS, etc.) are not presented.

Listing 1 Hashed DNS Zones Serialization Format

```
1: ; Zone: hszn.tld
2: ; Cuckoo Filter Parameters
3: buckets.hszn.tld      IN   TXT   <m>
4: entries.hszn.tld      IN   TXT   <b>
5: fgp-size.hszn.tld     IN   TXT   <f>
6: fgp-algo.hszn.tld     IN   TXT   <fgp()>
7: hash-algo.hszn.tld    IN   TXT   <hash()>
8: ; Cuckoo Filter Data
9: <n>.hszn.tld          IN   TXT   <RR Data>
```

The first part of Listing 1 (lines 2-7) includes details pertaining to the selected CF parameters and algorithms that were introduced in subsection 2.6.1.2. Specifically, the zone file involves information regarding the CF m, b and f values (lines 3-5). Moreover, details about the algorithms $fgp()$ and $hash()$ are also provided (lines 6-7). Each of the aforementioned details is dedicated an FQDN. FQDN prefixes are reserved words, properly selected to convey the appropriate meaning.

The second part of Listing 1 (lines 8-9) includes CF data. The straightforward approach to map CF fingerprints would be to assign each CF bucket to a separate RR of type TXT. However, such mapping method would have heavily increased the number of RR's, thus leading to oversized zone files: RR fields, i.e. FQDN's, TTL values and Type parameters introduce unnecessary bandwidth overhead during AXFR requests (see subsection 5.5.4). Instead, our schema

reduces the overall RR number by mapping the fingerprints of multiple CF buckets within a single RR of type TXT.

PAZM maps CF fingerprints in the HsZn's as hexadecimal numbers. All fingerprints are equally sized of $\lceil f/4 \rceil$ hexadecimal digits. Those requiring less than $\lceil f/4 \rceil$ hexadecimal digits are prepended with 0's.

Fingerprints are sequentially inserted within RR's until the maximum TXT-type RR value size, i.e. 255 Bytes for single strings [150], is reached. Buckets that are not full, i.e. contain less than b fingerprints, do not have explicit boundaries as the number of stored fingerprints varies between 0 and $b-1$. Thus, they are delimited from the next bucket with a dot. Conversely, full buckets do not require a trailing dot, hence memory consumption is further reduced. If the contents of a bucket do not completely fit within an RR, they are split and the remaining part is inserted in the next RR. FQDN prefixes begin from 0 and increase by 1 until all data are inserted.

Listing 2 depicts an HsZn RR that maps the first 25 buckets of a CF. Each CF bucket can accommodate up to 4 fingerprints and buckets with less than 4 fingerprints are delimited by dots. The fingerprint size is 12 bits, hence each requires 3 hexadecimal digits for mapping in the HsZn. Overall, 82 FQDN fingerprints are included in Listing 2, with the first fingerprint of each bucket underlined for clarity.

Listing 2 Hashed DNS Zone RR Data Example

```
0.hszn.tld IN TXT "c64.1dd4d1d590bfbf3ddaa20
3f6cb764b2c647a7063faff67fac8811df81c0fbe65f2.a5a.de2
bcd4666b6f10ba60e5cdc824ee3ba1807bd26d08a3.745a2f8
9e.395cbb723310f27e51c28ee3a96ad2e788092d2514513.44
33be06ed3314bc570ce85c921f5a59e07ee8db11.5f766e444e
96504eb01d090cc0d445.3eb."
```

5.4.3 Incremental DNS Zones (IncZn's)

These zones map the name changes of PltZn's. Therefore, administrators may be based on IncZn's to incrementally update their Filtering Modules without downloading the corresponding HsZn again. This is possible by performing IXFR requests since their most recently checked zone serial number.

Listing 3 depicts the serialization format of IncZn's. Each zone involves two parameters (lines 2-4): (i) *last-serial* indicates that PltZn modifications prior to this value are incorporated in the corresponding HsZn, whilst (ii) *sequence* increments each time the CF parameters are altered, e.g. when the filter is full and needs to be reshaped. Thus, *last-serial* defines the starting point for administrators to begin retrieving information from an IncZn and *sequence* defines if the related HsZn is stale and must be downloaded again.

Each zone RR (Listing 3, line 6) is related to a single name update and includes (i) the fingerprint fgp of the hashed FQDN as a hexadecimal number, (ii) the action associated with this

Listing 3 Incremental DNS Zones Serialization Format

```
1: ; Zone: inczn.tld
2: ; Zone Parameters
3: last-serial.inczn.tld        IN      TXT     <serial>
4: sequence.inczn.tld           IN      TXT     <seq-no>
5: ; Updates
6: <n>.inczn.tld  IN  TXT  "<fgp> <action> <h1>,<h2>"
```

fingerprint, i.e. *add* (addition) or *del* (deletion) and (iii) the pair of CF buckets h_1 and h_2 corresponding to the hashed FQDN. Notably, RR FQDN prefixes begin from 0 and increase by one to include each update.

5.5 Evaluation

Our experiments assess the HsZn privacy-awareness, the HsZn bandwidth consumption during AXFR requests and appropriateness of CF's for HsZn management. These are ultimately affected by the selected CF False Positive (FP) ratio. In a nutshell, large FP ratios result into (i) smaller probability of exposing HsZn contents, (ii) smaller CF fingerprint sizes and thus, less bandwidth consumption for HsZn transfers and (iii) larger volumes of NXDOMAIN responses by Authoritative DNS Servers.

In the following, we compare the effectiveness of CF's versus BF-based approaches. We configure a targeted FP probability of 0.3% for CF's and BF's. As CF's yield their memory advantages over BF's when they are almost full, we experiment with CF's that have 90% of their entries occupied; these values lead to bucket sizes that may hold up to 4 fingerprints, a common selection in CF's [19]. These parameters define 12-bit fingerprints (subsection 2.6.1.2) and thus, 3 hexadecimal characters are required to map each fingerprint in the HsZn. In contrast, BF's require 3 hash functions [16, 91] to provide a fair comparison against the CF-based approach. Notably, the Partial-key Cuckoo Hashing technique described in subsection 2.6.1.2 was adapted accordingly [151, 152] to support CF's whose bucket number is not limited to powers of 2.

5.5.1 Testbed Overview

We performed our experiments within our laboratory testbed. We used a VM comprised of 2 vCPU's and 16 GB RAM. The Hypervisor was a Dell PE R730 with Intel Xeon E5-2620 v3 2.4 GHz. We selected BIND 9 [25] for our Authoritative DNS Server as it is a common option among DNS administrators.

5.5.2 Dataset Description

Access to zone files is typically restricted. However, we gathered datasets for experimentation from servers within our campus and publicly available resources. Specifically, in April 2020,

we collected the zone files of:

- **ntua.gr**: We obtained the names of this zone using AXFR requests within our NTUA campus zone. This zone involves 8,294 distinct FQDN's.
- **se**: Names of this zone were obtained via an AXFR request as its contents are publicly available [114]. This zone includes 1,387,690 distinct FQDN's.
- **su & ru**: The contents of "su" and "ru" were exposed for a short time period in 2017 due to misconfigurations in the corresponding Authoritative DNS Servers [113]. These zones include 109,719 and 5,325,231 FQDN's respectively. Although these data may be old, they are comparable with the current zone sizes as reported by [7].

5.5.3 Hashed DNS Zones Privacy-Awareness

Although CF's store their elements hashed, attackers may still attempt to gain insight into zone contents. This is possible by performing brute force attacks, i.e. looking up all possible character combinations.

In the following, we assess the capabilities of CF's to withstand brute force attacks in the context of DNS. To that end, we fed a CF with all the permitted name combinations varying the first FQDN label between 3 and 7 characters. These consist of 37 characters: Latin alphabet, decimal digits and hyphen (not valid as an initial FQDN character). Note that these combinations include Internationalized Domain Names (IDN's) [153] in our dataset zones.

1st Label Length (Characters)	TP's (FQDN's)	FP's (FQDN's)	FP's/TP's (Ratio)
3	320	57	0.18
4	640	1,789	2.80
5	1,178	68,296	57.98
6	1,183	2,532,293	2,140.57
7	1,363	93,665,989	68,720.46

Table 5.1: CF privacy properties for DNS

Table 5.1 depicts the results of our assessment for a CF mapping our campus "ntua.gr" zone. Specifically, Table 5.1 includes the number of True Positives (TP's), i.e. matches for FQDN's stored in the CF, the number of False Positives (FP's), i.e. matches for FQDN's not included in the CF, and their ratio.

Attackers may guess FQDN's with first label length of 3 and 4 characters resulting into a relatively small number of FP's. FQDN's with prefixes longer than 5 characters are protected against brute force attacks with high certainty. Such attacks require exponentially longer time as the prefix grows since the total number of required hashing operations is prohibitively large (approximately 100 billion for 7 characters). Longer FQDN prefix lengths result into more FP's, thus discovery of plaintext names is next to impossible. We opted for an FP ratio of 0.3% which is sufficient to safeguard privacy. Note that BF's of similar targeted FP ratio would have resulted in comparable TP and FP numbers at the expense of significantly more hashing operations.

Indicative Zone (Distinct FQDN's)	Information Serialization Format			Cuckoo Filters (Actual Size)
	Cuckoo Filter (Multiple Buckets / RR)	Cuckoo Filter (Single Bucket / RR)	Bloom Filter (Multiple Bytes / RR)	
ntua.gr (8,294)	26.77 KB	63.91 KB	41.86 KB	13.51 KB
su (109,719)	352.1 KB	876.1 KB	553.11 KB	178.58 KB
se (1,387,690)	4.36 MB	11.21 MB	6.86 MB	2.21 MB
ru (5,325,231)	16.78 MB	43.76 MB	26.34 MB	8.46 MB

Table 5.2: Hashed DNS Zones: Bandwidth consumption during AXFR requests

5.5.4 Hashed DNS Zones Serialization

In the following, we determine the applicability of diverse information serialization formats for mapping FQDN's into HsZn's. To that end, we triggered AXFR requests towards our Authoritative DNS Server for various DNS zone sizes and recorded their memory footprint.

We considered the following serialization formats: (i) a CF with multiple buckets mapped within each RR, (ii) a CF with a single bucket corresponding to each RR and (iii) a BF with each RR containing multiple Bytes as hexadecimal numbers. Table 5.2 depicts the bandwidth consumed during AXFR requests for the zones of subsection 5.5.2 and the serialization formats mentioned above. It also depicts the actual CF size, i.e. the space required to hold the zone names in memory.

We observe that mapping plaintext names using a CF with multiple buckets assigned to each RR outperforms the other options. Moreover, although the consumed bandwidth is almost twice that of the in memory CF size, the overhead is manageable for modern network links. Thus, the selected FP ratio of 0.3% may map diverse zone sizes without issues.

5.5.5 Hashed DNS Zones Management

The PAZM performs various operations for managing HsZn's. In the following, we compare the latency of indicative actions using both BF's and CF's to map PltZn's. These include: (i) initial creation of the respective data structure in memory by hashing and inserting all the PltZn FQDN's and (ii) updating the data structures by performing 1,000 deletions and 1,000 insertions. Unlike CF's that directly support deletions, BF's need to be rebuilt excluding the removed data. Fig. 5.2 depicts the latency of performing these operations for the "ru" zone.

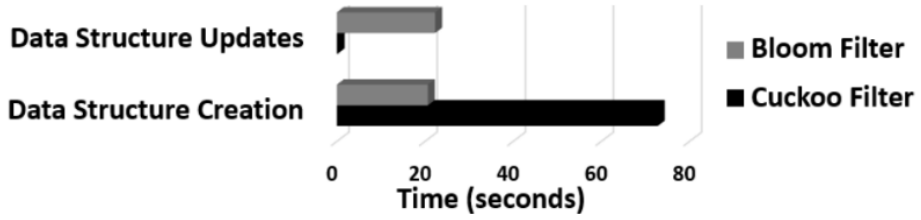

Figure 5.2: Latency of HsZn management operations

We observe that BF's are created significantly faster than CF's due to the element eviction process during insertions. However, CF's rapidly incorporate changes by employing incremen-

tal updates, contrary to BF's that need to be rebuilt as deletions are not supported. Thus, CF's clearly outperform BF's in DNS flood mitigation, where rapid updates are required; data structures are created only during the initialization phase, not affecting PAZM operations.

5.6 Summary & Concluding Remarks

We leveraged on probabilistic data structures to map the contents of Authoritative DNS Server zones in a privacy-aware format. This enables effective filtering of DNS traffic within Recursors or upstream scrubbing facilities. We employed CF's due to their time, space and dynamic deletion advantages over BF's. We evaluated our schema via experiments in our laboratory testbed. These indicated that our approach is promising for distributing Authoritative DNS Server zone names efficiently, while preserving privacy. Thus, DNS threat mitigation services may be distributed to third parties with no formal collaboration agreements.

This chapter focused on developing privacy-aware mechanisms for sharing whitelists of valid names pertaining to Authoritative DNS Server zones. However, DNS administrators may be unwilling to collaborate with third parties by implementing such mechanisms. Therefore, efficient anomaly-based methods, e.g. ML algorithms, may be required to enable effective DNS threat mitigation within Recursors, upstream cloud-based scrubbing infrastructures or other filtering appliances. Nevertheless, ML classification is usually implemented at the user space of the security mechanisms, thus mitigation throughput may be unable to cope with modern, high-speed DDoS attacks. In the following chapter, we will propose an XDP-based solution to accelerate ML inference within Recursors and effectively filter DNS DDoS attacks within the data plane.

Chapter 6

XDP-based Acceleration of Naive Bayes Classifier Inference for Efficient Data Plane DNS Attack Mitigation

In this chapter[1], we propose a schema that implements via Programmable Data Plane (PDP) methods efficient Machine Learning (ML) algorithms that differentiate between legitimate and DDoS attack traffic within cloud infrastructures. Specifically, we leverage on XDP to implement data plane Naive Bayes Classifier inference and effectively mitigate Water Torture attacks within data center Recursive DNS Servers (Recursors). DNS requests regarded as invalid by the Naive Bayes Classifier are dropped within the Linux kernel before any resources are allocated to them, while valid ones are forwarded to the user space to be resolved.

Our schema was assessed via a proof of concept setup within a virtualized environment. Learning and testing were performed via legitimate and malicious DNS data records whose statistical properties were consistent with datasets widely reported in the literature. Our experiments mainly focused on evaluating the filtering throughput of the proposed mitigation schema given the constraints imposed by XDP. We conclude that our XDP-based Naive Bayes Classifier significantly decreases the volume of attack traffic within the data plane, thus efficiently safeguarding Recursors.

6.1 Motivation

Programmable Data Planes (PDP's) [76] emerge as a promising alternative that facilitates Software Defined Networking and Network Function Virtualization (NFV) in cloud computing and data center infrastructures. As mentioned in Chapter 2, they enable the separation of data, control and management planes, while providing line-rate throughput for network services. Therefore, performance-critical tasks can be offloaded within the data plane, while complex actions are performed within the user space that constitutes a slower processing path [154].

Among various data plane programmability methods, eXpress Data Path (XDP) [10] has

[1]This work is included in the proceedings of the 10th IEEE International Conference on Cloud Networking (IEEE CloudNet 2021)

recently attracted significant attention. XDP enables the execution of extended Berkeley Packet Filter (eBPF) programs within the NIC driver level. Ingress traffic is delivered to the eBPF program that is executed on a per packet basis before any memory resources are allocated to the received packets. Therefore, XDP establishes high-performance data paths within the kernel, while maintaining the modules and the flexibility provided by the Linux kernel.

The aforementioned advantages render XDP suitable for various use cases, including DDoS attack mitigation [127]. As eBPF programs are executed within the Linux kernel, a verifier (eBPF verifier) [10] guarantees that these programs will not endanger kernel stability by restricting their functionality. Specifically, unbounded loops are forbidden, the program size is limited, while decimal numbers are not directly supported. These restrictions induce challenges to DDoS attack mitigation, particularly when Deep Packet Inspection (DPI) and/or ML techniques are required. DPI typically requires parsing the payload of ingress packets, which is of variable length, while ML inference is based on calculations, usually involving decimal values.

As mentioned in previous chapters, DDoS attack vectors that typically depend on DPI actions for their mitigation are those related to DNS Water Torture [4]. This attack was initially reported in 2016 during the Dyn cyber attack incident and is associated with the Mirai botnet [101]. Water Torture attack packets are appropriately crafted, i.e. by randomizing the first label of requested names, so that these are not contained in the zones of victim Authoritative DNS Servers. Moreover, names are never repeated by attackers, hence DNS caches of intermediary Recursive DNS Servers (Recursors) do not filter attack requests. Note that such attacks are mitigated more efficiently within Recursors, i.e. closer to the attack sources, based on whitelists of valid domain names or ML methods.

In this chapter, we propose a schema that employs PDP's to efficiently mitigate DNS Water Torture attacks targeting data center infrastructures. Our proposed solution leverages on XDP and eBPF to accelerate ML inference within the data plane of Recursors. Specifically, we implement in-network Naive Bayes Classifiers [98] tailored to the constraints of the eBPF verifier. Our classification process drops invalid DNS requests entirely within the Linux kernel, therefore decreasing the packet delay when attack traffic is forwarded to the user space to be filtered. Naive Bayes Classifiers are selected as simple, accurate and effective algorithms for mitigating Water Torture attacks, proposed and evaluated in [5].

The remainder of this chapter is structured as follows: In Section 6.2 we discuss related work; Section 6.3 provides a baseline description and lists the design principles of our proposed schema; Section 6.4 includes implementation details related to our approach; Section 6.5 involves the experimental evaluation of our suggested mechanism. Finally, in Section 6.6 we conclude our work.

6.2 Related Work

XDP and eBPF have been recently used to enable Software-Defined Networking and NFV functionality. In [154] a hybrid architecture is introduced that differentiates execution environments

between a fast path (data plane) and a slow path (user space). Simple, but time-critical tasks are performed in the fast path using eBPF/XDP, while complex tasks that cannot be implemented in the data plane are executed in the slow path. Moreover, XDP and eBPF are employed to efficiently chain Virtual Network Functions (VNF's). Similarly, the Polycube [155] and Mizar [156] projects employ eBPF/XDP to implement lightweight VNF's.

In [157], Decision Tree Classifiers were implemented using eBPF to efficiently analyze network traffic. This ML method, implemented within the data plane, resulted into significant performance improvements compared to the corresponding user-space packet monitoring solution. Our eBPF/XDP-based mitigation schema follows a similar approach.

A straightforward method of mitigating Water Torture attacks involves rate limiting ingress DNS requests [158]. To avoid dropping excessive amounts of requests triggered by legitimate Internet users, ML approaches were proposed to implement selective filtering of attack requests. In [5] Naive Bayes Classifiers were assessed as accurate and effective methods to distinguish between valid and invalid domain names. However, their proposed implementation relied on specialized hardware for efficient packet handling, therefore increasing complexity and reliance on specific hardware platforms. Alternatively, [126] employed Random Forests (RF's) to filter attack traffic. Nevertheless, selected features included information related to the source IP address of DNS requests, thus rendering this work vulnerable to IP spoofing.

Other approaches for mitigating Water Torture attacks rely on whitelists of valid domain names. Specifically, [159] filters attack requests by constructing lists from the successfully resolved domain names within DNS caches of Recursors. However, such filtering approaches may result in dropping newly registered or infrequently requested names. Note that in previous work (Chapters 3 and 4), we employed probabilistic data structures, i.e. Bloom Filters (BF's) [16] to map Authoritative DNS Server zones, thus creating name whitelists. This solution accomplished time and space efficient Water Torture mitigation, while retaining the privacy of zone contents. However, complete contents of DNS zones may not be accessible to Recursors unless Authoritative DNS Server administrators are willing to collaborate by adopting privacy-aware mechanisms for securely sharing their zones (see Chapter 5).

Our proposed schema implements XDP-based Naive Bayes Classifiers to accelerate ML feature extraction and classification within the data plane of Recursors. Contrary to [5], our schema is a software-based solution that can be applied on general purpose hardware and thus, it is suitable to be offered as a VNF within cloud infrastructures. Moreover, we revised the features utilized in [5] by considering Domain Generation Algorithm (DGA) traffic filtering approaches [64]: Similarly to Water Torture, DGA's involve randomized names, hence features used for DGA traffic detection may also be applied in Water Torture attack mitigation.

6.3 Proposed Schema: Overview & Design Features

In this section, we provide a high-level description of our proposed schema and discuss its main design principles.

6.3.1 High-Level Description

Fig. 6.1 depicts an overview of our schema for the mitigation of Water Torture attacks within the data plane of Recursors. Ingress traffic is detained by an XDP Hook attached on the Recursor NIC driver level. Subsequently, this traffic is delivered to an eBPF Program that is executed on a per-packet basis. Initially, this program filters packets to detect DNS requests. These are kept for further inspection, while the rest of the traffic, i.e. DNS responses or traffic unrelated to DNS, is forwarded to the User Space through the Kernel Network Stack. Afterwards, the program parses the name included in the DNS request and calculates the values of the Naive Bayes features corresponding to the first DNS label, i.e. prefix, of the name. Based on these values, Naive Bayes classifies names as valid or invalid. Names categorized as invalid are dropped within the Linux kernel, while those labeled valid are forwarded to the User Space for name resolution. Filtering of malicious packets is performed entirely within the data plane, therefore increasing the overall packet processing throughput. Invalid packets are not forwarded to the User Space and thus, they are not delayed by the Kernel Network Stack.

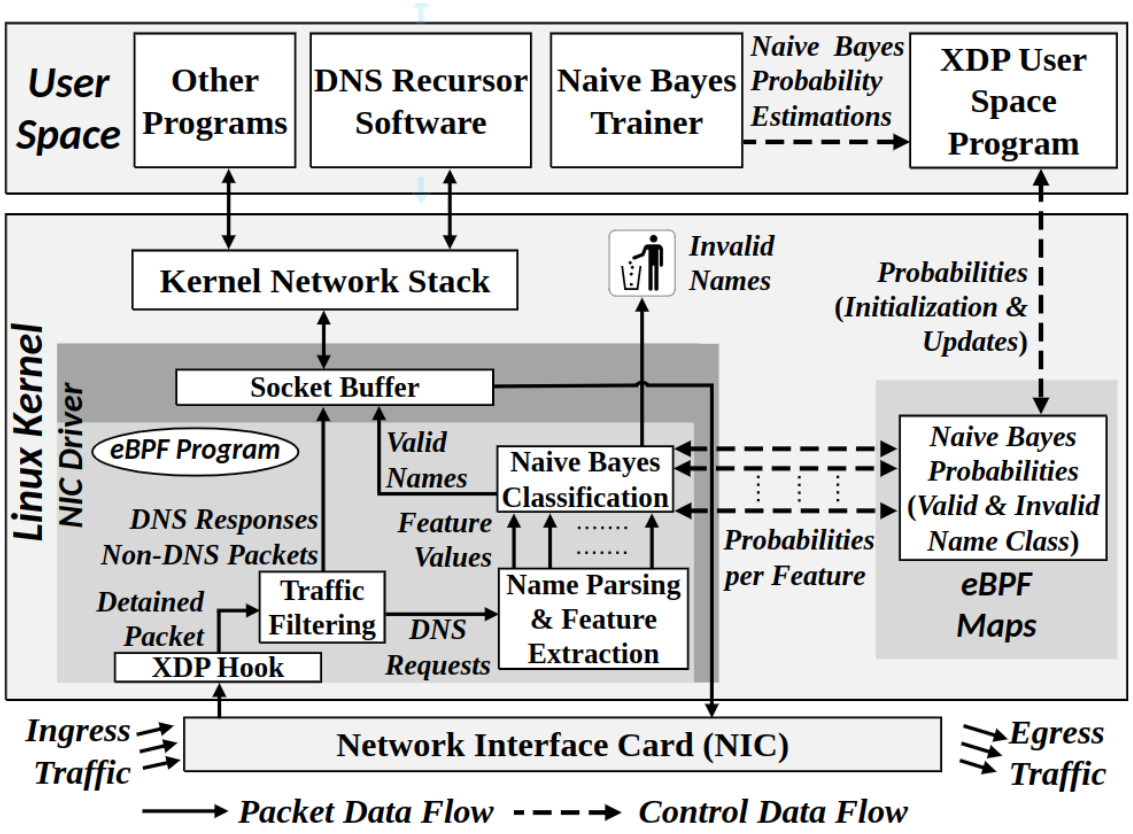

Figure 6.1: Baseline design of our proposed schema

The classification decisions made by Naive Bayes are based on the probabilities estimated per feature and name class, i.e. valid and invalid, during the training phase of the algorithm. Estimation of these probabilities is performed within the User Space by the Naive Bayes Trainer. When the training phase is completed, the probability estimates are delivered to the XDP User Space Program. This program is responsible for: (i) invoking the eBPF verifier to guarantee the

safety of the Linux kernel, (ii) attaching the XDP Hook at the NIC driver level of the Recursor, (iii) loading the eBPF Program for execution and (iv) creating and populating the eBPF maps with the estimated probabilities per feature and name class. The eBPF maps are in-kernel, key-value data stores that enable value exchanges among eBPF programs and the XDP User Space Program. During execution of the eBPF program, Naive Bayes retrieves from eBPF maps the estimated probabilities per feature and name class to label name prefixes as valid or invalid.

6.3.2 Design Principles

The main design principles of our schema are the following:

- **Efficient, in-network ML feature extraction and inference**: Our proposed schema leverages on PDP's to provide high-throughput packet filtering of DDoS attack requests. Specifically, after a training phase, an XDP-based Naive Bayes Classifier efficiently extracts features from requested names and assesses their validity. Therefore, packets are classified as valid or invalid entirely within the data plane of Recursors and the user space DNS software is not required to process excessive attack traffic.
- **DNS request filtering that does not rely on lists of valid names**: Our schema employs ML to categorize ingress DNS requests based on features extracted from their associated names (subsection 6.4.3). Therefore, our solution may complement whitelist-based approaches for Water Torture mitigation when lists of valid names are not available. Moreover, our schema stores only the feature conditional probabilities required by Naive Bayes Classifiers, instead of whitelists that may include millions of names [7], thus decreasing the overall storage and memory requirements.
- **NFV-compliant mitigation schema**: Our approach successfully separates data, control and management planes, hence adhering to the Software-Defined Networking paradigm. Thus, our solution may be deployed as a VNF in cloud environments, while centralized management of deployed services is possible.
- **DDoS attack mitigation independent of specialized hardware**: We use an open-source software solution, i.e. eBPF/XDP, that depends only on the version of the available NIC drivers. Contrary to hardware-based data plane methods, e.g. FPGA's and P4 switches [12], XDP relies on C-like software implementation independent of specific hardware platforms.

6.4 Implementation Details

In this section, we elaborate on details regarding the implementation of our schema. Specifically, we discuss (i) restrictions of implementing our schema using XDP and eBPF, (ii) provide information about the diverse programs developed and (iii) describe the features selected for our Naive Bayes Classifier. Our implementation is available from [160].

6.4.1 eBPF/XDP Restrictions

The eBPF programs are executed within the Linux kernel. Specifically, the eBPF verifier guarantees that these programs will not damage kernel stability, e.g. by performing out of bounds memory accesses. The main restrictions imposed by the eBPF verifier include: (i) forbidding unbounded loops, (ii) constraining the eBPF program size and (iii) not supporting decimal numbers directly. In the following, we provide details on the aforementioned restrictions tailored to the development of our XDP-based mitigation schema:

6.4.1.1 Absence of Unbounded Loops

Parsing DNS names typically depends on unbounded loops as their length varies and thus, their position in DNS messages is not clear until they are completely parsed. However, such loops are not permitted in eBPF. To that end, we used a bounded loop that is unrolled up to a predefined name length (subsection 6.4.1.2).

6.4.1.2 Limited eBPF Program Size

DNS names may consist of up to 255 characters [1]. These lengths violate the maximum program size permitted by the eBPF verifier. We accordingly restricted the maximum name length to a smaller value allowed by the eBPF verifier, i.e. 200 characters. Thus, names with length over 200 characters cannot be completely parsed by our eBPF program. However, this restriction does not reduce the accuracy of our classification process as it is performed on the first label, i.e. the prefix, of DNS names. Recall that Water Torture involves names whose prefix is randomly generated. According to [1], the maximum length of DNS labels is 63 characters, thus prefixes of names with length over 200 characters will be definitely parsed in our eBPF program.

6.4.1.3 Lack of Decimal Numbers Support

ML algorithms, including Naive Bayes, typically require calculations involving decimal numbers, not supported by eBPF. Therefore, we multiplied Naive Bayes conditional probabilities estimated during the training phase by powers of 10 and retained only the integer part of these numbers for calculations within the eBPF program. Probabilities equaling 0 after enforcing the multiplication factor are set to 1, thus being ignored during Naive Bayes inference.

This process may affect classification accuracy. Multiplication by small factors, i.e. powers of 10, may lead to reduced classification accuracy when features are not represented accurately enough. Conversely, bigger multiplication factors may result in buffer overflows within the eBPF program during the multiplication of conditional probabilities, thus decreasing accuracy.

6.4.2 Details on Software Implementation

Our implementation consists of the following three programs: (i) the Naive Bayes Trainer, (ii) the XDP User Space Program and (iii) the eBPF program. In the following, we provide details

on the aforementioned programs.

6.4.2.1 Naive Bayes Trainer

This program estimates the probabilities that will be used by Naive Bayes for classifying ingress DNS requests into valid/invalid based on the provided training examples. Afterwards, these probabilities are multiplied by an appropriate multiplication factor (see subsection 6.4.1.3) and provided to the XDP User Space Program. We implemented the Naive Bayes Trainer in Python.

6.4.2.2 The XDP User Space Program

This program acts as a workflow engine for the eBPF program and is executed within the user space of Recursors. It is responsible for (i) invoking the eBPF verifier, (ii) attaching the XDP hook, (iii) loading the eBPF program if permitted by the eBPF verifier and (iv) delivering the Naive Bayes estimated probabilities to the eBPF program by initializing and filling the appropriate eBPF maps. We implemented this program in Python leveraging on the BPF Compiler Collection (BCC) toolset [161].

6.4.2.3 The eBPF Program

This program is responsible for filtering attack traffic within the data plane of Recursors when Water Torture attacks are detected. The eBPF program is executed per ingress packet at the Recursor. Initially, traffic is filtered to differentiate between DNS requests, i.e. UDP segments destined towards port 53, and traffic irrelevant to Water Torture attack mitigation, i.e. DNS responses or packets not related to DNS. Then, the eBPF program parses the prefix of the name included in the DNS request question section via a bounded loop and extracts the features of Naive Bayes.

After feature extraction, the eBPF program retrieves the Naive Bayes probabilities, estimated during the training process, from the appropriate eBPF maps using feature values as keys. This is performed for both Naive Bayes classes, i.e. one related to valid names and another related to invalid names. Finally, the eBPF program calculates and compares the two products required for classification; one derived from the multiplication between the conditional probabilities of features associated with the class of valid requests and one corresponding to the class of invalid requests. Requests perceived as benign are forwarded to the user space, while requests involving names classified as invalid are dropped within the kernel, thus offloading the Recursor. The eBPF program was implemented in a restricted version of the C programming language, which complies with the eBPF verifier restrictions.

6.4.3 Selected Features

The features of our Naive Bayes Classifier were selected based on their ability to differentiate between meaningful and randomized strings. Specifically, we adopted features used in detecting

botnet traffic by DGA's [64]. Our selected features for evaluating prefixes of DNS names as valid or invalid are summarized in Table 6.1.

Feature Notation	***Feature Description***
F_1	Prefix length
F_2	Number of digits
F_3	Length of maximum digit sequence
F_4	Number of consonants
F_5	Length of maximum consonant sequence
F_6	Number of vowels
F_7	Length of maximum vowel sequence

Table 6.1: Selected features for Naive Bayes Classifier

Note that using a small number of features may significantly decrease classification accuracy when they are not carefully selected. However, an excessive number of features may result in buffer overflows within eBPF programs when Naive Bayes conditional probabilities are multiplied (subsection 6.4.1.3).

6.5 Evaluation

This section presents experiments aimed to evaluate our proposed schema. Initially, we describe the datasets employed in our experiments (subsection 6.5.1). Subsequently, we focus on the evaluation of our eBPF program (subsection 6.5.2). Finally, we assess the accuracy of our selected features and compare them with those used in related approaches [5] (subsection 6.5.3).

6.5.1 Dataset Selection

Our schema was evaluated using two separate datasets: Dataset I of name prefix lists that were used to train our Naive Bayes Classifier and assess its accuracy; Dataset II of DNS request packet captures that were used to evaluate our eBPF program. The datasets were retrieved/constructed in July 2021.

Dataset I, used to train and evaluate Naive Bayes, consisted of two subsets: (i) A list containing valid DNS prefixes extracted from the top names of an index including the most popular Internet names [162] and (ii) a list containing invalid DNS prefixes generated using a custom Python script with prefix lengths between 10 and 30 DNS characters as in [5]. Each subset contained 800K prefixes; 700K prefixes were used for the training of the Naive Bayes Classifier and the remaining 100K for testing its accuracy. Notably, both training subsets included the same number of prefixes, hence prior probabilities were equal.

Dataset II, used to evaluate our eBPF program, comprised of two packet capture (pcap) files: (i) the Legitimate Traces involving 100K DNS requests generated using the top 100K prefixes of an Internet name popularity list [163] and (ii) the Water Torture Traces involving 4 million DNS requests generated based on invalid names produced by 5 DGA's (CoreBot, Monero Downloader, newGOZ, Reconyc, Qadars) [164]. The average length of prefixes included in Water

Torture Traces was 24 characters.

6.5.2 Evaluation of our eBPF Program

In the sequel, we describe the testbed used for our experiments (subsection 6.5.2.1), investigate how mapping decimal numbers within the data plane affects the accuracy of our eBPF program (subsection 6.5.2.2) and assess the filtering throughput of our schema under a Water Torture attack scenario (subsection 6.5.2.3). Note that we define invalid DNS requests correctly classified by our schema as True Positives (TP's), while successfully identified valid DNS requests as True Negatives (TN's).

6.5.2.1 Testbed Description

Our schema was tested via a proof of concept setup within a virtualized environment. Our setup consists of the following 3 Virtual Machines (VM's):

- **Traffic Generator:** A VM that forwards Water Torture and Legitimate Traces to the Recursor at specified packet rates. It consists of 4 vCPU's and 4 GB RAM.
- **Recursor:** A VM that forwards DNS requests towards the Authoritative DNS Server using BIND 9 [25]. Our eBPF/XDP based mitigation solution is deployed within this VM. The Recursor comprises of 4 vCPU's and 4 GB RAM, running Linux kernel version 4.15.
- **Authoritative DNS Server:** A VM that responds to Recursor requests. It involves a zone mapping the names of the Legitimate Traces. This VM is implemented in BIND 9 and consists of 2 vCPU's and 2 GB RAM.

6.5.2.2 Accuracy of Mapping Decimal Numbers within eBPF

The purpose of this assessment is to investigate how mapping decimal numbers within eBPF programs affects the in-network Naive Bayes accuracy in filtering DNS requests. Recall that the Naive Bayes probabilities estimated during the training phase cannot be directly supported by eBPF since decimal numbers are not available. Thus, all estimated probabilities are multiplied by a power of 10 (multiplying factor) and only the integer part of decimal numbers is retained, thus sacrificing accuracy (see subsection 6.4.1.3).

We utilized the Traffic Generator to forward the Legitimate Traces (Dataset II) to the Recursor, while our eBPF/XDP based mitigation solution was active. Fig. 6.2 depicts the percentage of valid DNS requests that are correctly classified by the Naive Bayes Classifier of our eBPF program (TN rate). We applied three different multiplying factors, i.e. 100, 1,000 and 10,000, before loading the estimated probabilities within eBPF maps of our program. Our maps were able to store up to 128-bit integers. Naive Bayes was trained using Dataset I and the features described in subsection 6.4.3.

We observed a notable accuracy loss when small or large multiplying factors were considered. Small multiplying factors, e.g. 100, decreased the accuracy of Naive Bayes probabilities

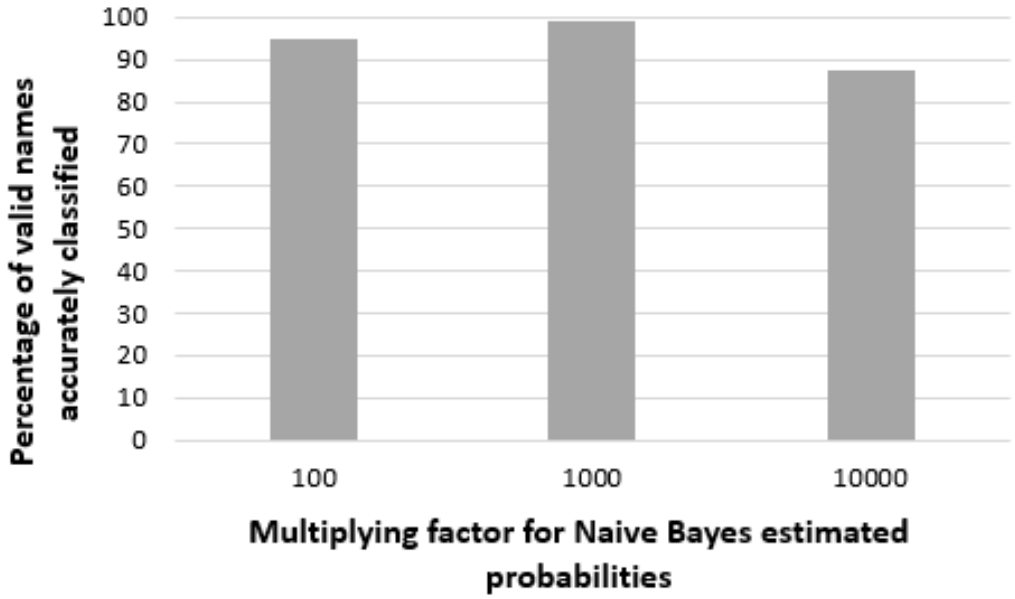

Figure 6.2: Investigation of diverse multiplying factors for mapping decimal numbers within eBPF/XDP

representation, therefore leading to an evident packet loss of valid DNS requests. On the contrary, large multiplying factors, e.g. 10,000, significantly reduced classification accuracy since buffer overflows frequently occurred within the eBPF program. We conclude that 1,000 is a satisfactory multiplication factor by successfully forwarding 99.01% of valid names. This value will be used in the experiments of subsection 6.5.2.3.

Note that a straightforward approach to support more accurate decimal number representations is to define eBPF maps that can handle integers of bigger sizes; recall that maps in our assessment held values of up to 128 bits. However, defining such maps will deteriorate the filtering capabilities of our schema as modern CPU architectures are optimized for integer values of up to 64 bits.

6.5.2.3 Filtering Throughput of the Proposed Schema

The purpose of this experiment is to evaluate the capabilities of our proposed eBPF/XDP based schema in filtering DNS requests during a Water Torture attack. Specifically, we employed the Traffic Generator to simultaneously send Water Torture and Legitimate Traces (Dataset II) to the Recursor, which forwards them to the Authoritative DNS Server. Legitimate traces were forwarded at a rate of 1 Kpps, while Water Torture Traces at a rate of 40 Kpps. The rate of the Legitimate Traces was based on statistics from packet captures performed in [165] within Recursors of campus networks. The rate of the Water Torture Traces was selected as high enough to downgrade the quality of the DNS service offered by the Recursor.

Fig. 6.3 depicts the percentage of valid DNS requests (TN rate) forwarded by the Recursor to the Authoritative DNS Server during a Water Torture attack when (i) mitigation is not applied and (ii) mitigation is performed within the Recursor using our eBPF/XDP based solution.

We observe that our eBPF/XDP based mitigation solution significantly improved the Recur-

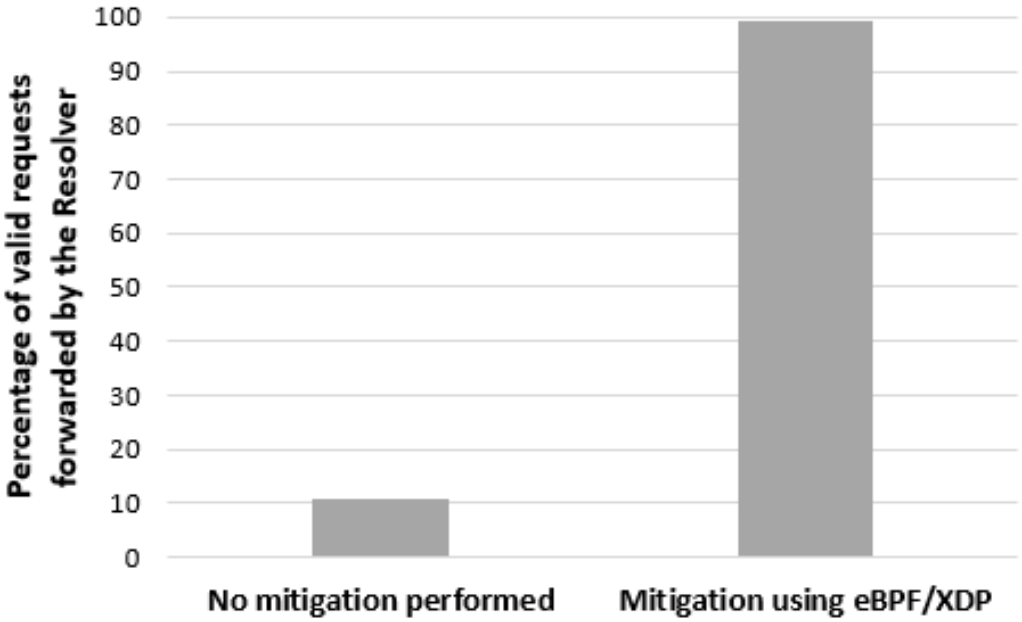

Figure 6.3: Number of valid DNS requests forwarded by the Recursor

sor filtering capabilities. When mitigation was not applied, the Recursor forwarded only 10.7% of valid requests, while name resolution was impacted by the massive number of invalid requests. However, when our solution was used, the Recursor filtered the vast majority of invalid DNS requests (TP's), hence forwarding the 98.95% of valid requests (TN's).

6.5.3 Evaluation of Selected Features

The purpose of this assessment is to verify if our selected features F_1 to F_7 (subsection 6.4.3) are sufficient for accurately differentiating between valid and invalid domain names. To that end, we compare our features with those selected in [5] to mitigate Water Torture attacks; recall that both our schema and [5] depend on Naive Bayes Classifiers to categorize ingress traffic. Specifically, regarding information included in the DNS name prefix that characterizes Water Torture attacks, [5] relied on two features: (i) the length of the name, i.e. feature F_1 also employed in our work, and (ii) the summation of all Naive Bayes estimated probabilities for each ordered pair of adjacent characters, i.e. bigram.

Fig. 6.4 depicts the percentage of inaccurately classified valid and invalid name prefixes when three diverse feature combinations for Naive Bayes are used: (i) our selected features (F_1 up to F_7), (ii) 8 features consisting of our selected features (F_1 up to F_7) plus the bigrams feature as in [5] and (iii) prefix length (F_1) feature plus the bigrams feature as in [5]. Naive Bayes Classifiers were trained and assessed using Dataset I.

We concluded that our 7 selected features are suitable for classifying name prefixes as valid or invalid; our features misclassify only 1.92% of valid prefixes and 1.27% of invalid ones. Moreover, our features outperformed those of [5], while adding the bigrams feature does not improve accuracy.

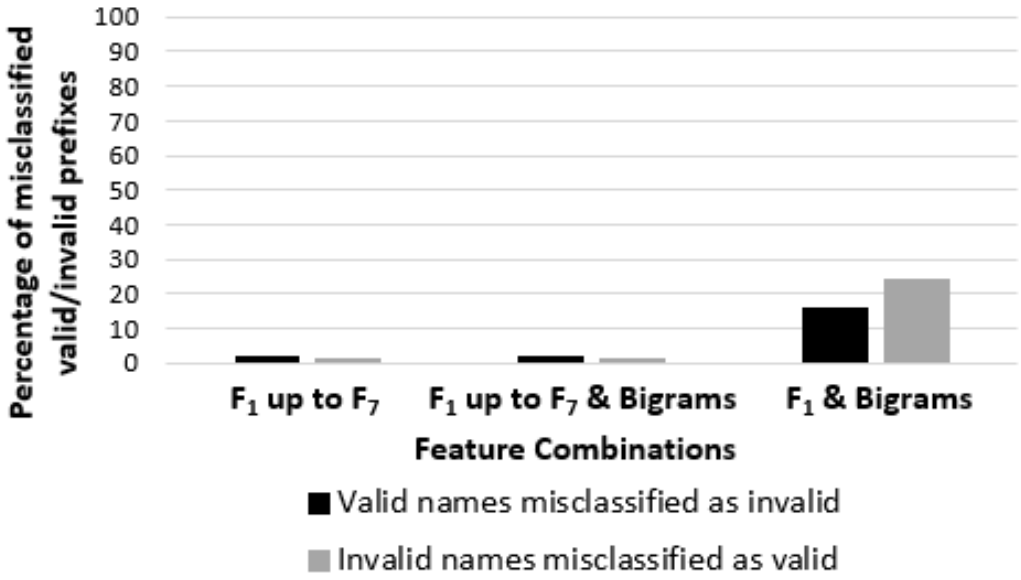

Figure 6.4: Evaluation of diverse feature combinations

6.6 Summary & Concluding Remarks

We relied on data plane programmability to effectively mitigate DNS Water Torture attacks within Recursors. Our schema leveraged on XDP and eBPF to implement Naive Bayes Classifiers within the data plane and efficiently differentiate between valid and invalid DNS requests. Our evaluation demonstrated that our schema significantly improved filtering throughput of Recursors, thus relieving the user-space DNS service from processing excessive amounts of invalid requests.

ML algorithms have been widely employed to effectively filter DNS cybersecurity threats, including DDoS attacks and DGA traffic. Apart from Naive Bayes, which was used in this chapter, tree-based ML classifiers and deep neural networks have also been successfully utilized for accurately differentiating between benign and malicious domain names. However, despite the promising results of the aforementioned ML algorithms, DNS administrators may be unwilling to deploy ML classifiers within their networks unless they receive appropriate justifications regarding their operation. In the following chapter, we will leverage on eXplainable Artificial Intelligence (XAI) techniques to provide model-agnostic interpretations of DGA traffic classifiers.

Chapter 7

SHAP Interpretations of DNS Classifiers for Analyzing DGA Family Characteristics

Domain Generation Algorithms (DGA's) have been employed by botnet orchestrators for controlling infected hosts (bots), while evading detection by performing multiple DNS requests, mostly for non-existing domain names. With blacklists ineffective, modern DGA filtering methods rely on Machine Learning (ML). Emerging needs for higher intrusion detection accuracy lead to complex, non-interpretable black-box classifiers, thus requiring eXplainable Artificial Intelligence (XAI) techniques.

In this chapter[1], we utilize SHAP to derive model-agnostic, post-hoc interpretations on DGA name classifiers. This method is applied to binary supervised tree-based classifiers (e.g. eXtreme Gradient Boosting - XGBoost) and deep neural networks (Multi-Layer Perceptron - MLP) to assess domain name feature importance. SHAP visualization tools (summary, dependence, force plots) are used to rank features, investigate their effect on model decisions and determine their interactions. Specific interpretations are detailed for identifying names belonging to common DGA families pertaining to arithmetic, wordlist, hash and permutation based schemes. Learning and interpretations are based on up-to-date datasets, such as Tranco for benign and DGArchive for malicious names. Domain name features are extracted from dataset instances, thus limiting time-consuming and privacy-invasive database operations on historical data.

Our experimental results demonstrate that SHAP enables explanations of XGBoost (the most accurate tree-based model) and MLP classifiers and indicates the characteristics of specific DGA schemes, commonly employed in attacks. We envision that XAI methods will expedite ML deployment in networking environments where justifications for black-box models are required.

7.1 Motivation

ML algorithms have been widely employed within the cybersecurity domain for effectively filtering massive amounts of data and classifying malignant traffic. Such algorithms have been commonly used in the field of botnet traffic detection and for classifying names originating

[1]This work is included in IEEE Access (Volume 11)

from DGA's [6]. Tree-based ML classifiers and deep neural networks are utilized to differentiate between legitimate and malicious DNS names with promising accuracy results.

Development of DGA name classifiers has been motivated by the desire for ML models of higher performance. Therefore, simple and intrinsically explainable ML classifiers have been replaced by complex, black-box models that are not interpretable. Thus, developers are incapable of understanding their models to debug them and assert their intended operation, while users cannot receive justifications on model decisions made on their data. Finally, regulators are unable to ensure that models deployed within critical infrastructures comply with GDPR [99] or equivalent legislations.

The aforementioned limitations led to investigations for XAI techniques [13] to provide interpretations (and possibly explanations) on ML model operation. As mentioned in [14], post-hoc and model-agnostic XAI algorithms are typically preferred. Post-hoc algorithms are applied to ML models after learning is completed; model-agnostic ones are independent of the selected ML models, e.g. tree classifiers and neural networks. Explanations may be (i) *global* detailing model behavior on entire sets of sample points and (ii) *local* reporting how models make classification decisions for specific inputs. A promising post-hoc and model-agnostic approach is SHAP [15, 166], which is capable of global and local explainability.

Our work leverages on XAI to analyze the operation of binary, supervised DGA name classifiers that distinguish between legitimate and malicious[2] names, thus detecting botnet traffic abusing DNS. We train and evaluate various tree-based classifiers (Random Forests - RF's, Gradient Boosting - GB, eXtreme Gradient Boosting - XGBoost, Adaptive Boosting - AdaBoost, Extremely Randomized Trees - ExtraTrees) and a deep neural network (Multi-Layer Perceptron - MLP). SHAP is subsequently employed to determine and compare the classification criteria of XGBoost [167], which was the most accurate tree model, and MLP deep neural network [96] in a post-hoc and model-agnostic manner. Our experimental analysis focuses on global and local model interpretations used to rank the impact of utilized features and indicate how their individual values contribute to classification decisions. Relying on multiple SHAP visualization tools (i.e. summary, dependence and force plots [13, 166]) we investigate how the developed models (i) differentiate between benign and malicious domain names and (ii) identify which features have the most significant contribution in classifications of names originating from well-known fundamental DGA generation schemes that produce malicious names [6]. Learning and interpretations are based on linguistic and statistical features, directly extracted from domain names included within up-to-date datasets of benign and malignant DNS names.

Our main contributions are summarized as follows:

- SHAP interpretations of DGA name classifiers based on deep neural networks (MLP's) and comparison of their decision-making criteria versus tree-based ML models (XGBoost).
- Identification of dominant features utilized for malicious domain name detection pertaining to specific DGA generation schemes (arithmetic, wordlist, hash and permutation based).

[2]Throughout this chapter, DNS names are considered malicious if they are produced by DGA's. Non-DGA names, even those related to malignant activities (e.g. malware propagation), are labeled as benign names in the training set.

- Extraction of linguistic and statistical features leading to accurate and real-time classification of DGA names with no reliance on time-consuming and privacy sensitive external repository operations.
- Training and interpretations based on the most updated and inclusive dataset of DGA names, i.e. the DGArchive repository [6, 20] including 105 DGA families.
- Open-sourced implementation available from our GitHub repository [168].

The remainder of this chapter is structured as follows: Section 7.2 summarizes related work; Section 7.3 provides a high-level overview of our methods used for interpreting DGA name classifiers; Section 7.4 elaborates on implementation details pertaining to our approach; Section 7.5 includes our experimental results and interpretations of DGA name classifiers based on XGBoost and MLP. Finally, in Section 7.6 we conclude our work.

7.2 Related Approaches & Key Contributions

This section outlines related research approaches (subsection 7.2.1) and details our key contributions (subsection 7.2.2).

7.2.1 Related Work

Various approaches have been proposed for the detection of DGA names with promising results, e.g. [58–68, 169, 170]. However, the aforementioned approaches emphasize on improving detection accuracy, but they do not deliver global and local model and feature interpretations.

Interpreting DGA name classifiers has recently attracted significant interest. In [171] neural network classifiers are interpreted based on their weights. A system for result visualization is also presented to facilitate model comprehension. However, interpretations rely on model-specific XAI methods applicable exclusively to deep learning models, whilst the total features are limited for visualization purposes. In [172] multi-class DGA name classifiers are developed based on features directly extracted from domain names and feature importance is assessed using various statistical methods. Nevertheless, [172] is limited to global explainability of DGA classifiers, thus neglecting model interpretations on specific DNS names. Moreover, the effect of different DGA schemes on model decisions is not addressed.

In [173–175] SHAP and/or equivalent XAI techniques (e.g. Local Interpretable Model-Agnostic Explanation - LIME [176] and Counterfactual Explanations [177]) are employed to provide global and local interpretations on binary DGA name classifiers. Although the aforementioned approaches deliver promising results, they are limited mainly to tree-based ML classifiers. These approaches focus on interpreting how names are classified as benign or malicious, therefore neglecting how the characteristics of different DGA families affect classification decisions. Furthermore, feature calculation in [173] and [175] requires resource-intensive operations on databases involving historical data, e.g. IP reputation lists, WHOIS lookups and TTL values from DNS responses. These are usually time-consuming and may raise privacy concerns.

7.2.2 Key Contributions

Our approach relies on SHAP for model-agnostic (regardless of the selected models) and post-hoc (after the learning procedure is completed) validation of DGA name classifier operation. Our models are based on features extracted entirely from given names, hence resource-intensive operations on privacy-sensitive historical DNS data are not required. We compare interpretations derived from tree-based models (i.e. XGBoost) and neural networks (i.e. MLP's) using both global and local explanations. Notably, we extend related approaches by analyzing how binary classifier feature rankings perform when facing diverse DGA schemes, e.g. following testing methods used in use cases related to radio communications and health systems [178, 179]. Finally, malicious DNS data used for training and interpreting our models are selected from DGArchive; we included 105 DGA families, a significantly higher number compared to [173–175].

7.3 Proposed Schema: Overview & Design Features

This section outlines our analysis principles (subsection 7.3.1) and provides a baseline description of our schema for developing and interpreting DGA name classifiers (subsection 7.3.2).

7.3.1 Design Principles

The main design principles of our approach are:

- **Model-agnostic ML interpretations**: We employ the SHAP KernelExplainer [100] to interpret our DGA name classifiers independently of the underlying ML model. Thus, we analyze the operation of tree-based and deep neural network classifiers in a unified manner.
- **Local and global interpretations**: Our approach relies on SHAP to rank feature contributions in classification decisions made on specific input instances for local explainability and lists of domain names for global explainability.
- **Analysis relying on various SHAP visualization tools**: Multiple SHAP visualization methods (i.e. summary, dependence and force plots) are employed to estimate feature importance, determine how feature values affect model decisions and investigate feature interactions.
- **Classification based on domain-specific features**: ML models are trained on features directly extracted from DNS names without requiring costly database operations on historical data that may raise privacy concerns. Such features conceive the statistical and linguistic properties of DNS names, hence they are suitable for real-time DGA name classifications.
- **Explanations for diverse DGA schemes**: We assess the effect of different DGA family properties on feature contributions. This way we infer how the binary DGA name classifiers distinguish between legitimate and malicious DNS names for specific DGA schemes (i.e. arithmetic, wordlist, hash and permutation based).

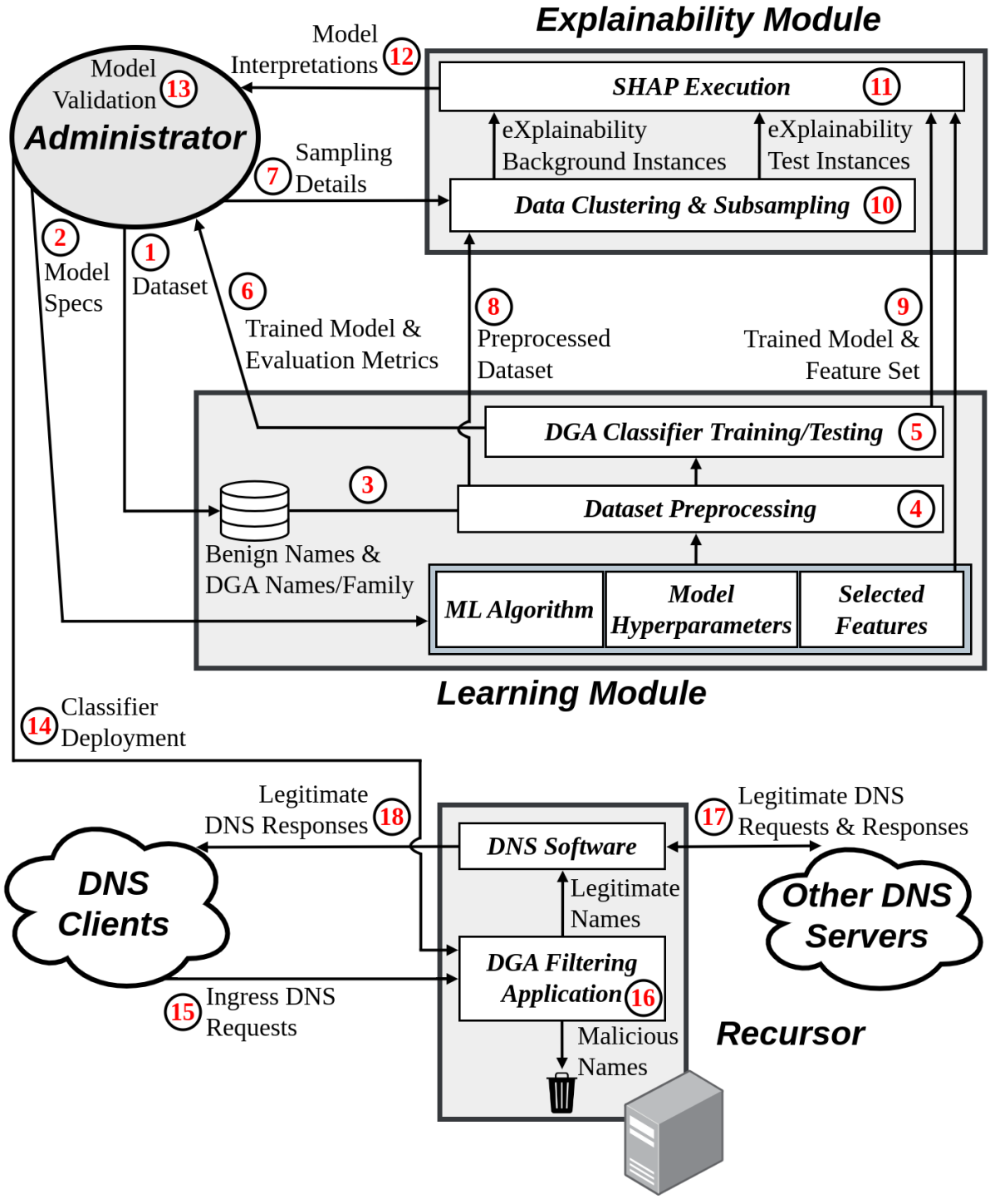

Figure 7.1: Baseline design

7.3.2 Baseline Design

Fig. 7.1 depicts an overview of our approach for DGA traffic detection based on accurate and reliable classifiers. The purpose of the Administrator is to train supervised binary classifiers that effectively differentiate between benign and DGA names, validate their dependable operation via XAI techniques (specifically SHAP) and deploy filtering rules to drop botnet traffic.

The architecture of Fig. 7.1 consists of three components:

- **Learning Module**: Data are preprocessed and the necessary learning parameters are defined to train and evaluate DGA name classifiers.
- **Explainability Module**: SHAP is used to analyze and validate the operation of name classifiers developed by the Learning Module.
- **Recursor**: Ingress DNS requests are inspected using the trained DGA name classifiers; those involving malicious names are dropped, while legitimate DNS traffic is forwarded for name resolution.

The Administrator initially selects the learning dataset that will be utilized for tuning DGA name classifiers (step 1). The selected data consist of benign and malicious (i.e. DGA generated) DNS names labeled for binary classification purposes. Malicious dataset labels include the DGA algorithm used for name construction; such information is typically available from reverse engineering efforts on DGA malware installed within infected hosts [164].

Details of the Learning Module operation are subsequently determined (step 2). The Administrator defines the model specifications required for tuning name classifiers, i.e. the ML algorithm, the model hyperparameters and the selected features. The learning dataset is then retrieved (step 3) and preprocessed (step 4) based on the selected features and ML model details. The DGA Classifier is subsequently trained and evaluated (step 5), while assessment results and tuned model parameters are returned to the Administrator (step 6).

Upon completion of the learning phase, the Administrator configures the Explainability Module by determining the reduced dataset instances required for SHAP execution (step 7). This step refers to the clustering and subsampling processes required for keeping the SHAP running time within feasible time periods. In steps 8 and 9 the Learning Module feeds the trained DGA Classifier, the selected features and the preprocessed dataset to the Explainability Module. This dataset is then clustered and subsampled (step 10) to derive the instances required for SHAP; the eXplainability Background Instances (XBI's) used in SHAP calculations for assessing feature importance and the eXplainability Test Instances (XTI's) consisting of the input sampling points used to eventually derive model interpretations. Note that, in our case XTI's were subsampled from the class of malignant DGA names since our purpose was to assess feature importance per DGA generation scheme.

After SHAP analysis is completed (step 11), the Explainability Module provides the Administrator with global and local model-agnostic interpretations of the trained classifiers (step 12). The Administrator gathers the Learning and Explainability module results to validate model op-

eration (step 13). If the classifier accuracy and explanations are satisfactory, the Administrator deploys appropriate DGA filtering procedures within the Recursor (step 14).

In step 15, ingress DNS requests from DNS Clients are inspected by the Recursor (step 16). Malicious DNS requests are dropped, whereas legitimate ones are resolved by the DNS Software, e.g. BIND [25], installed within the Recursor (steps 17 and 18).

7.4 Implementation Details

This section elaborates on feature selection (subsection 7.4.1), on the development and operations of the Learning Module (subsection 7.4.2) and on details pertaining to the Explainability Module (subsection 7.4.3).

7.4.1 Selected Features

We leverage on feature values that are directly extracted from given domain names and denote linguistic properties (e.g. values denoting the number of vowels) and statistical measures (e.g. entropy values). Such features facilitate real-time DNS traffic inspection and limit sensitive data exchanges by not requiring storage of privacy-sensitive information. As already stated, we do not employ historical data features (e.g. time-based patterns of DNS responses and IP reputation measures), which typically require excessive processing resources and storing them may raise privacy concerns [64].

Prior to feature extraction, valid DNS suffixes (one or multiple zone namespaces, e.g. "*com*" and "*gov.uk*") are removed from domain names as in [64]. These are not generated by DGA's, hence they are not meaningful to the learning process. Identification of valid DNS suffixes is based on the Mozilla public suffix list [180]. Note that removing these suffixes mapped multiple distinct names to common prefixes within the learning dataset, e.g. "google.com" and "google.fr" were both reduced to "google". As a result, classifiers are tuned towards accurately recognizing frequently requested DNS names; their appearance frequency within the dataset reflects specific trends of DNS queries resolved by Recursors.

The features used for DGA name classification are outlined in Table 7.1; feature selection was based on approaches available from the literature, e.g. [61, 64, 181]. In the following, features 44, 47, 48 and 50 are further analyzed:

- ***Vowel_Freq (feature 44):*** Determines the number of vowels included within the domain name, i.e. letters a, e, i, o, u and y; considering y as a vowel typically increases classification accuracy as reported in [67].
- ***Reputation (feature 47):*** Evaluates domain name *Reputation* defined as an indication of its legitimacy [182]; the higher the *Reputation* the more legitimate the name may appear. A method for measuring the reputation score of a domain name is the appearance frequency of N-grams (i.e. sequences of N consecutive characters) present in benign names and absent in malignant ones [183]. Estimating *Reputation* requires a preprocessing stage whereby a

Number	Feature Name(s)	Description
1	Length	Length of the domain name
2	Max_DeciDig_Seq	Length of maximum decimal digit sequence
3	Max_Let_Seq	Length of maximum letter sequence
4 - 29	Freq_A, ..., Freq_Z	Frequency of letters A-Z within the name
30 - 39	Freq_0, ..., Freq_9	Frequency of digits 0-9 within the name
40	Spec_Char_Freq	Number of special characters (hyphens, dots) within the name
41	Ratio_Spec_Char	Fractional division of Spec_Char_Freq and Length
42	DeciDig_Freq	Number of decimal digits (0-9) within the name
43	Ratio_DeciDig	Fractional division of DeciDig_Freq and Length
44	Vowel_Freq	Number of vowels within the name
45	Vowel_Ratio	Fractional division of Vowel_Freq and Length
46	Max_Gap	Length of the longest name label
47	Reputation	Number of whitelisted N-grams (N = 3, ..., 7)
48	Words_Freq	Number of concatenated meaningful words within the name
49	Words_Mean	Average length of concatenated meaningful words obtained from feature 48
50	Entropy	Shannon Entropy of the name

Table 7.1: Selected Features for DGA name classification

whitelist is constructed based on the N-grams derived from a set of legitimate DNS names (e.g. the Tranco list [21]). *Reputation* of a given domain name is evaluated by determining how many of its N-grams are included in the aforementioned *whitelist*. N values are selected between 3 and 7 characters as in [183]; unigrams (i.e. N = 1) and bigrams (i.e. N = 2) are excluded because most of them exist in both legitimate and malicious names, thus affecting the learning process and hindering feature importance.

- ***Words_Freq (feature 48):*** Determines the number of meaningful words within the given names. Words are extracted using the Wordninja Natural Language Processing (NLP) tool [184] similarly to [185]. Wordninja probabilistically splits strings into concatenated words based on the unigram frequency of words appearing within the English Wikipedia. As in [186], words shorter than 3 characters (e.g. pronouns and articles) are ignored as their effect to the learning process is not significant.
- ***Entropy (feature 50):*** Estimates domain name randomness using Shannon Entropy [64]. We used the standard definition of entropy:

$$H(X) = -\sum_{x \in X} p(x) log_2 p(x)$$

where X is the set of characters included within a DNS name and $p(x)$ the frequency of character $x \in X$.

7.4.2 Learning Module

This module trains and evaluates supervised binary classifiers that differentiate between legitimate and DGA names. The labeled dataset comprised of benign and malicious names is retrieved and the Learning Module proceeds with dataset preprocessing by performing feature extraction. Pairwise feature correlations are calculated using the Pearson's Correlation Coefficient (PCC)

statistical measure [187] to detect redundant features not contributing significantly to the learning process. Upon detecting pairs with PCC's exceeding a predefined threshold, a feature is randomly selected and evicted from the dataset, eventually accelerating the learning process without significant performance degradation.

The resulting dataset is randomly split into the training set (used for tuning the binary classifier) and the testing set (used for evaluating model generalization). Training and testing instances are scaled between 0 and 1 using Min-max normalization based on minimum and maximum values of training instances as in [188]. The Learning Module completes dataset preprocessing by balancing the number of benign and malicious class instances. Training set instances are oversampled using the Synthetic Minority Over-sampling Technique (SMOTE) [189], similarly to [188]. SMOTE synthetically generates instances following training set statistical properties to reduce imbalance between given classes.

Finally, the Learning Module trains and evaluates DGA name classifiers. We trained tree-based classifiers (i.e. Random Forest - RF, Gradient Boosting - GB, eXtreme Gradient Boosting - XGBoost, Adaptive Boosting - AdaBoost, Extremely Randomized Trees - ExtraTrees) and a deep neural network (i.e. Multi-Layer Perceptron - MLP). Tree classifiers were developed using scikit-learn [190] and XGBoost Python Package [191], whereas MLP's with Keras [192]. Model hyperparameters were fine-tuned using Grid Search, which exhaustively explores a subset of the ML algorithm hyperparameter space and selects the best performing classifier [193].

7.4.3 Explainability Module

This module analyzes the operation of DGA name classifiers using SHAP, eventually delivering global and local model-agnostic post-hoc interpretations to the Administrator.

The preprocessed dataset, the trained model and the selected features are initially retrieved from the Learning Module. The preprocessed dataset is then clustered and subsampled to limit SHAP analysis within reasonable time constraints [14]. The eXplainability Background Instances (XBI's) are obtained as the centroids of K-means clustering on the training set, whereas eXplainability Test Instances (XTI's) are derived by randomly subsampling the testing set. The XBI's are used to tune SHAP values and the XTI's to interpret decisions made by the DGA name classifiers.

Subsequently, SHAP KernelExplainer [100] is used to derive global and local interpretations by ranking features according to their contribution in classification decisions and determining interactions between them. SHAP offers various visualization tools to facilitate comprehension of interpretations [14, 166]. We relied on the following SHAP plots:

- **Summary plots**: Features are ranked in descending order according to their impact on model decisions. XTI's are mapped as instance dots based on their positive or negative contributions to model classifications, i.e. their SHAP values depicted in the horizontal dimension. Low and high values of features are additionally mapped on summary plots to depict their effect on classifier operation. SHAP relies on a color palette to distinguish

feature values; extreme values are visualized using a pair of basic colors (e.g. blue and red), whereas basic color shades denote their intermediary values.

- **Dependence plots**: They demonstrate contributions of specific features on model decisions. XTI's are mapped as dots on a two-dimensional plot; the horizontal axis includes all possible values of an investigated feature, whereas the vertical axis depicts the corresponding SHAP values, i.e. their impact on model decisions. Dependence plots also visualize the correlation between the investigated feature and an additional one that mostly interacts with it. This interacting feature is determined by evaluating the joint effect of all possible feature pairs, therefore estimating their influence on classification accuracy using the Shapley interaction values [13,194]. Low and high values of the interacting feature are depicted using the aforementioned color palette, thus facilitating conclusions of how feature interactions jointly affect classification decisions.
- **Force plots**: They demonstrate feature contributions on specific XTI's (typically single local instances). A pair of basic colors is used to discern model features according to whether they contribute positively (e.g. red) or negatively (e.g. blue) to classification decisions. Names and values of features mostly contributing to model decisions are included in the plot, whereas less important feature names and values are omitted. A decimal number (denoted with bold characters) corresponds to the final result returned by the binary classifier.

7.5 Evaluation

This section includes our experimental analysis. Subsection 7.5.1 describes the selected dataset and subsection 7.5.2 outlines the experimental testbed. Subsection 7.5.3 involves the Learning Module performance evaluation that assesses the binary DGA name classifier accuracy. Finally, subsection 7.5.4 includes the SHAP interpretations extracted by the Explainability Module.

7.5.1 Datasets

ML models were evaluated using malicious and benign domain names, typically used for building DGA name classifiers. Our data were retrieved in Spring 2023.

Malicious DNS names were obtained from DGArchive [20], a moderated repository continuously updated with DGA names resulting from reverse engineering efforts on DGA malware code. We retrieved roughly 200 million domain names corresponding to 105 distinct DGA families pertaining to all generation schemes (i.e. arithmetic, wordlist, hash and permutation based). The total repository size and constraints of our experimental infrastructure rendered training of DGA name classifiers time-consuming and memory intensive. Therefore, we sampled DGArchive and randomly extracted 10,000 DNS names from each DGA family as in [195]; families involving less than 10,000 names were included without subsampling. Eventually, our dataset consisted of 600,775 DGA names, which were used to train, evaluate and interpret DGA

name classifiers.

Legitimate DNS names were selected from Tranco [21], a public online service ranking domain names based on their popularity. Tranco merges data from various name ranking services, namely Alexa, Cisco Umbrella, Majestic and Farsight. Name rankings are calculated over long time periods (e.g. 30 days), thus mitigating the impact of abrupt daily fluctuations and/or list manipulation attempts. However, Tranco still contains a small percentage of DGA names that are frequently requested by large numbers of infected Internet devices (bots). Therefore, we filtered the Tranco dataset [196] by removing names included within DGArchive; these amounted to 0.57% of Tranco entries. We subsequently utilized the top-ranked 1 million entries from the remaining Tranco names similarly to [174]. Following [183] we used the first 100,000 to construct the *whitelist* pertaining to the *Reputation* feature (subsection 7.4.1); the remaining 900,000 were used to train and assess the DGA name classifiers.

The aforementioned name sets were labeled as benign and malignant without indicating specific families of malicious DGA names. Binary classifiers were selected instead of multi-class ones. Although multi-class classifiers may provide insight in specific DGA families, they are typically less accurate than binary ones in segregating benign and malignant names [58].

7.5.2 Testbed Overview

Experiments were performed within our laboratory infrastructure. We utilized a VM comprising of 8 virtual cores and 24GB physical memory. The hypervisor was a Dell PE R730 with Intel Xeon E5-2620 v3 2.4 GHz. Training of neural networks was accelerated using the NVIDIA GeForce GTX 1050 Ti 4GB [197] graphics card.

7.5.3 Learning Module

The Learning Module was evaluated by assessing (i) the pairwise correlation among selected features and (ii) the performance of supervised binary DGA name classifiers. Assessments were performed using the dataset of benign and malicious names described in subsection 7.5.1.

Pearson's Correlation Coefficient (PCC) was utilized to detect highly correlated features. PCC's were calculated for all feature pairs and those exceeding 0.9 (by absolute value) were considered strongly correlated [198]. In such feature pairs, a feature was selected at random and evicted from the dataset. In particular, *Ratio_DeciDig* was determined as strongly correlated to other features, hence it was removed from subsequent experiments.

We selected Random Forests (RF's), Gradient Boosting (GB), eXtreme Gradient Boosting (XGBoost), Adaptive Boosting (AdaBoost) and Extremely Randomized Trees (ExtraTrees) as indicative algorithms of tree-based classifiers; Multi-Layer Perceptrons (MLP) were selected as representative models of deep neural networks. Classifiers were trained and evaluated using the dataset described in subsection 7.5.1. This dataset was randomly split into two parts using the *train_test_split* method of scikit-learn [168]; 80% was utilized as the training set and the remaining 20% as the testing set.

Hyperparameters	Considered Values	Best Classifier Value
Random Forest - RF		
Number of Trees	10, 20, ..., 200	200
Maximum Tree Depth	10, 20, ..., 200	50
Gradient Boosting - GB		
Number of Trees	10, 20, ..., 100	100
Maximum Tree Depth	10, 20, ..., 50	20
eXtreme Gradient Boosting - XGBoost		
Number of Trees	10, 20, ..., 200	100
Maximum Tree Depth	10, 20, ..., 200	20
Adaptive Boosting - AdaBoost		
Number of Trees	10, 20, ..., 1000	520
Extremely Randomized Trees - ExtraTrees		
Number of Trees	10, 20, ..., 500	260
Other Parameters: Default scikit-learn values		

Table 7.2: Hyperparameter tuning of tree classifiers using Grid Search

Hyperparameters	Considered Values	Best Classifier Value
Number of Hidden Dense Layers	1, 2, 3	3
Neurons per Hidden Dense Layer	100, 200, 300	Dense Layer 1: 300 Dense Layer 2: 200 Dense Layer 3: 200
Dropout Probability	0.2, 0.5	0.2
Batch Size	256, 512	512
Epochs	100 epochs with EarlyStopping [199]	
Loss Function	BinaryCrossentropy [200]	
Optimizer	Adam [201]	
Activation Functions	Input/Hidden Layers: ReLU Output Layer: Sigmoid	

Table 7.3: Hyperparameter tuning of MLP classifier using Grid Search

Grid Search was used to tune model hyperparameters. The number and maximum depth of RF, GB and XGBoost trees were varied as described in Table 7.2. The number of AdaBoost and ExtraTrees estimators were varied as described in the table. Similarly, multiple MLP configurations were considered by varying the hidden layers number, the neurons per layer, the batch size and the rate of dropout regularization layers placed between the hidden layers to reduce overfitting. Considered MLP hyperparameters are described in Table 7.3.

Based on the accuracy of ML models, classifier performance was assessed as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

where True Positives (TP's) are the correctly classified DGA names, True Negatives (TN's) are the correctly categorized benign names, False Positives (FP's) are the incorrectly classified benign names and False Negatives (FN's) are the misclassified malicious names.

Grid Search determined that among RF, GB, XGBoost, AdaBoost, ExtraTrees and MLP classifiers the best accuracy scores on the testing set were 94.67%, 94.66%, 94.81%, 92.32%, 94.67% and 94.51% respectively[3] as shown in Table 7.4. Their configuration details are sum-

[3]Filtering repetitive name prefixes (see subsection 7.4.1) within the training and testing sets yielded comparable accuracy results, specifically

marized in tables 7.2, 7.3.

Algorithm	Best Classifier Accuracy
Random Forest - RF	94.67%
Gradient Boosting - GB	94.66%
eXtreme Gradient Boosting - XGBoost	94.81%
Adaptive Boosting - AdaBoost	92.32%
Extremely Randomized Trees - ExtraTrees	94.67%
Multi-Layer Perceptron - MLP	94.51%

Table 7.4: Accuracy of best classifiers

7.5.4 Explainability Module

The Explainability Module was evaluated based on SHAP interpretations derived on the trained models (subsection 7.5.3) for the dataset described in subsection 7.5.1. We investigated (i) the features used to discern benign and malicious names derived from multiple DGA families and, (ii) the most influential features utilized to differentiate specific DGA schemes.

Interpretations were derived for 105 DGA families of the DGArchive repository and are available from our GitHub repository [168]. However, for illustration purposes representative results are presented in this chapter for 4 indicative DGA families pertaining to 4 diverse DGA schemes (see subsection 2.3.2). Specifically, as in [202] results are presented for the following: (i) *DirCrypt* (arithmetic-based), (ii) *Matsnu* (wordlist-based), (iii) *Bamital* (hash-based) and (iv) *VolatileCedar* (permutation-based).

Similarly to [14], XBI's were selected as the cluster centroids resulting from K-means execution on the training set with K equal to 50. XTI's used for interpreting how name classifiers differentiate between benign and malicious names derived from all DGA families were obtained by randomly subsampling 250 DGA names from the testing set. Interpretations pertaining to specific DGA families were based on XTI's randomly subsampled from testing set entries of these specific families; families with less than 250 names were included without subsampling.

A greater number of XBI's and XTI's yielded in our extensive experiments insignificant interpretation improvements, while SHAP running time increased dramatically [168]. Using the aforementioned parameters, the Learning Module and the Explainability Module required approximately 2 days to complete their operation.

The following subsections present SHAP interpretations for XGBoost (which was the most accurate tree-based model) and the MLP deep neural network model. Interpretations are based on multiple SHAP plots: (i) summary plots pertaining to 250 XTI's from all DGA families (subsection 7.5.4.1), (ii) summary plots involving XTI's from selected DGA families (subsection 7.5.4.2), (iii) dependence plots pertaining to 250 XTI's from all DGA families (subsection 7.5.4.3), (iv) dependence plots including XTI's from specific DGA families (subsection 7.5.4.4) and (v) force plots for selected domain names (subsection 7.5.4.5). Legitimate and malicious name classes are denoted with numbers *0* and *1* respectively. Thus, negative SHAP values contribute to benign name classifications, whereas positive values to DGA name classifications.

94.39% for XGBoost (best tree-based classifier) and 94.31% for the MLP neural network. Thus, we did not consider filtering them in our experiments pertaining to the Explainability Module.

7.5.4.1 XGBoost & MLP Classifier Summary Plots for all DGA Families

In this subsection SHAP summary plots are used to explain the operation of binary DGA name classifiers. Fig. 7.2 and 7.3 demonstrate respectively XGBoost and MLP classification criteria for segregating malicious names from benign ones. Analysis was based on 250 XTI's, illustrated as colored dots in the horizontal dimension, from all DGA families. In these summary plots blue color is used to denote low feature values, whereas red color is utilized for high feature values (see subsection 7.4.3).

Fig. 7.2 depicts the 20 most influential features used by the XGBoost binary classifier. The most effective features are *Reputation*, *Length*, *Freq_Q*, *Words_Mean*, *Words_Freq* and *DeciDig_Freq* ranked in order of descending importance. High *Length* and *DeciDig_Freq* values favor malicious name classifications. Such behavior is related to lengthy names and high decimal digit frequencies, typically employed by most DGA's to avoid coincidence with legitimate registered domain names. As expected, high *Reputation* and *Words_Freq* values mostly point to benign name categorizations since the presence of many whitelisted N-grams and meaningful words are linked to legitimate names. *Max_DeciDig_Seq* contribution is significantly smaller compared to the impact of the aforementioned features; it is ranked 12th in terms of contribution to classification decisions. Finally, high feature values of *Words_Mean* may inconclusively affect both benign and malicious name classifications.

Fig. 7.3 shows that the most influential features used by the MLP classifier are *Reputation*, *Length*, *Max_DeciDig_Seq*, *Words_Mean* and *DeciDig_Freq*. Similarly to XGBoost, MLP relies predominantly on *Reputation* and *Length* features. *Max_DeciDig_Seq* was the 3rd most important feature for MLP with higher values pointing to benign name classifications. Recall that for XGBoost, *Max_DeciDig_Seq* was ranked 12th, a much lower significance level (Fig. 7.2). Likewise, *Vowel_Freq* feature significantly affects MLP decisions ranking as the 8th most influential feature, while XGBoost dependence on *Vowel_Freq* is not even among the 20 most significant features of Fig. 7.2. This may be partially explained by the difference of XGBoost and MLP in modeling learning tasks. The former mainly relies on splitting training set instances based on dominant feature deviations; following boosting methods strong tree estimators are eventually constructed by iteratively improving weaker classifiers. The latter (MLP) tunes its weights during back propagation towards directions that linearly combine feature values, forming induced local fields that are further subjected to non-linear activation functions (e.g. ReLU, Sigmoid). Thus, XGBoost mainly relies on boosting methods based on significant feature deviations [97], while MLP on weighted feature differences.

7.5.4.2 MLP Classifier Summary Plots for Selected DGA Families

This subsection addresses explanations pertaining to binary MLP classifiers tested for XTI's derived from specific DGA families. In Fig. 7.4, 7.5, 7.6 and 7.7 we respectively present summary plots for 4 DGA families selected from 4 different generation schemes: (a) *DirCrypt* (arithmetic-based), (b) *Matsnu* (wordlist-based), (c) *Bamital* (hash-based) and (d) *VolatileCedar*

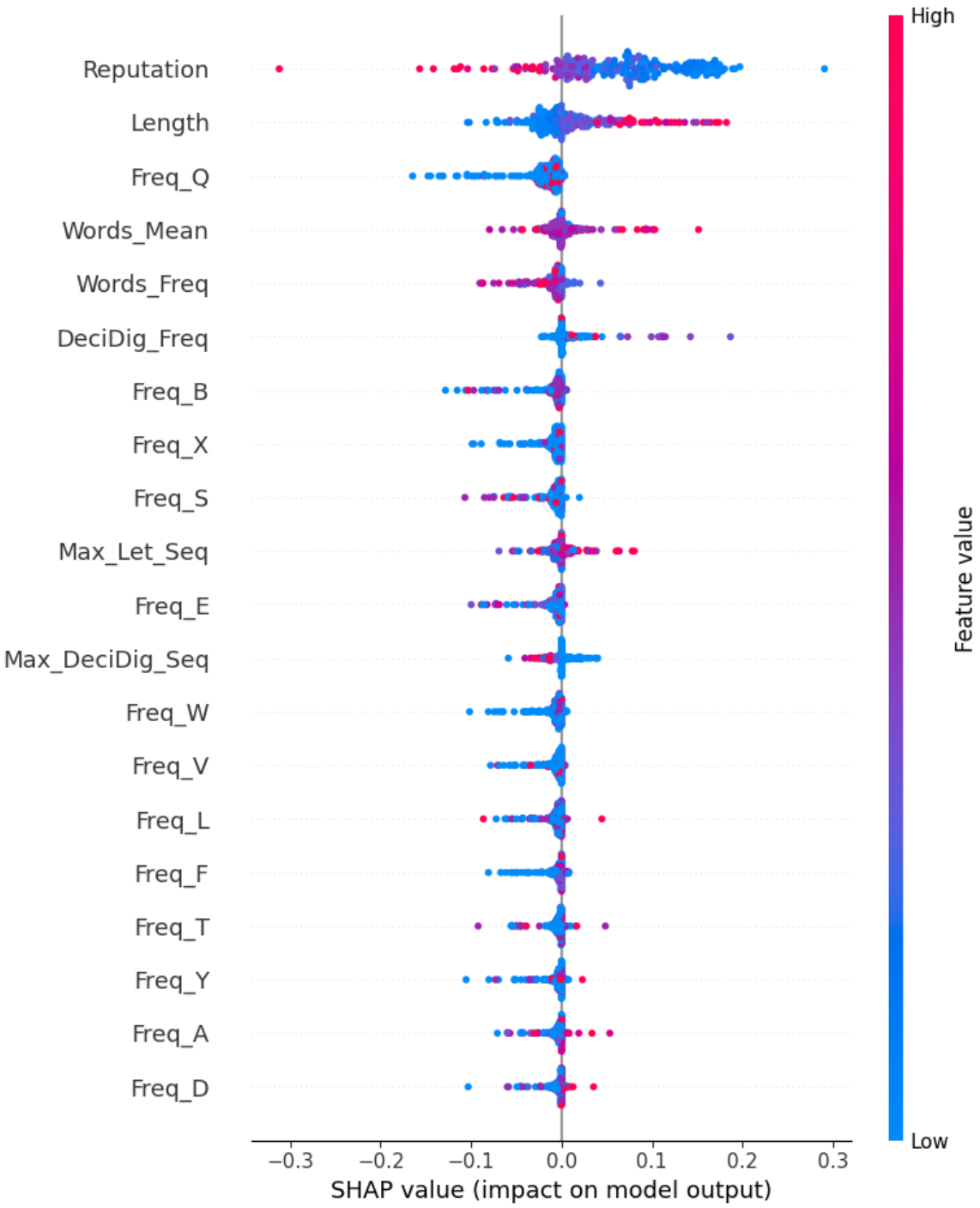

Figure 7.2: SHAP summary plot on XTI's including malicious names from all DGA families (the utilized algorithm is eXtreme Gradient Boosting - XGBoost)

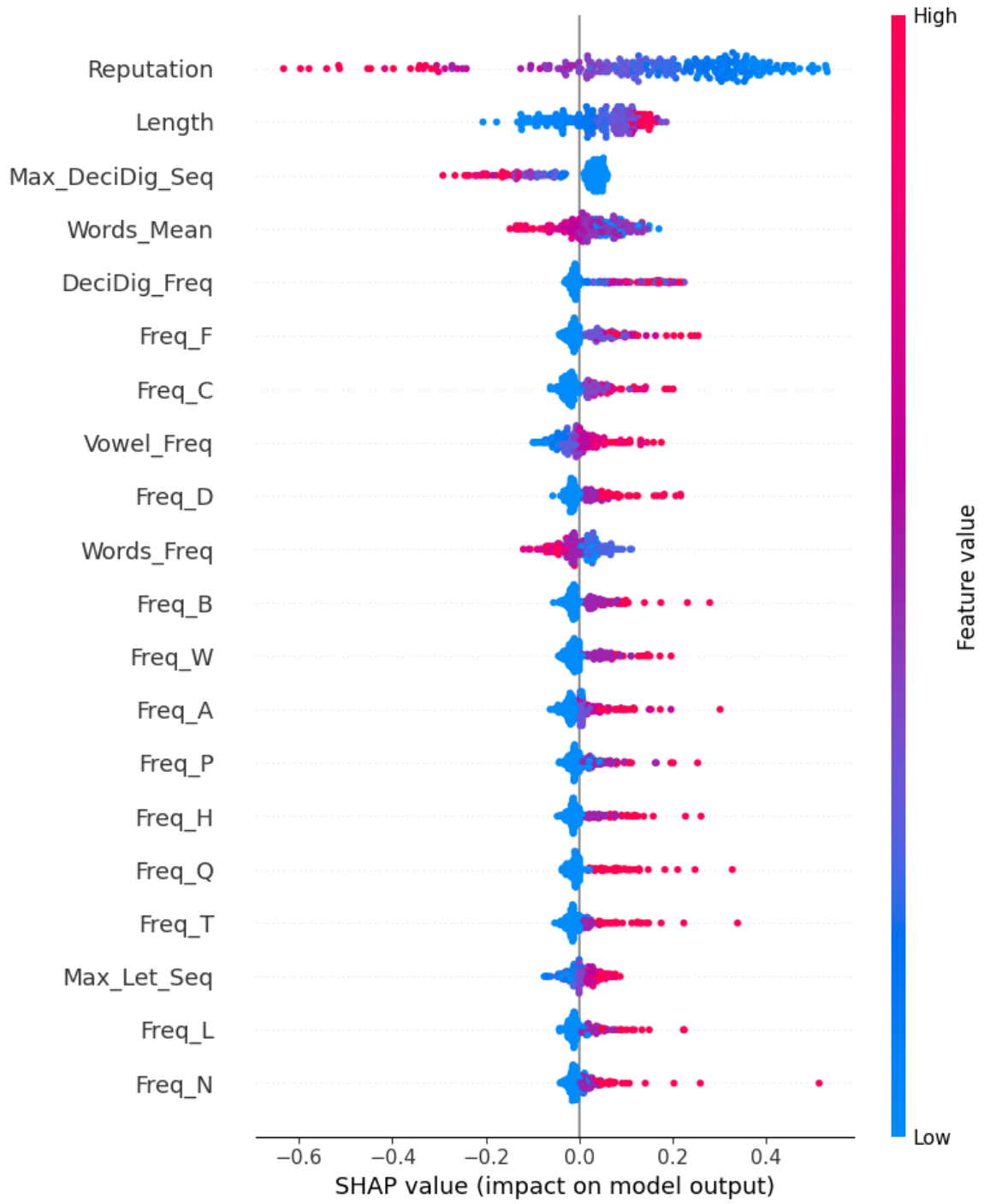

Figure 7.3: SHAP summary plot on XTI's including malicious names from all DGA families (the utilized algorithm is Multi-Layer Perceptron - MLP)

DGA Family	Scheme	Indicative Name (Prefix)
DirCrypt	Arithmetic	iwqvktutvmptevjbnzy
Matsnu	Wordlist	chickenpriceresearch
Bamital	Hash	b7a8b33957a2f95105353aa1873aebda
VolatileCedar	Permutation	shplayergetadobaefl

Table 7.5: Indicative names per DGA family

(permutation-based). In Table 7.5 we list four indicative malicious names pertaining to each of the aforementioned DGA families; note that typical suffixes, e.g. "com" and "info", are not included in the table. These schemes and their respective families have the following properties [6]:

- **Arithmetic-based DGA's (e.g. *DirCrypt*):** Domain names are generated by concatenating randomly selected characters. *DirCrypt* is based on the 26 English alphabet letters to produce names between 8 and 20 characters. Names typically contain long consonant sequences and are characterized by increased randomness compared to benign names.
- **Wordlist-based DGA's (e.g. *Matsnu*):** Random dictionary words are concatenated to generate malicious domain names resembling legitimate ones. *Matsnu* forms long names between 12 and 24 characters by joining multiple dictionary words of relatively short length [203].
- **Hash-based DGA's (e.g. *Bamital*):** They rely on the hexadecimal representation resulting from hashing domain names. *Bamital* is based on MD5 hash function to generate names consisting of 32 hexadecimal digits.
- **Permutation-based DGA's (*VolatileCedar*):** Multiple DGA names are produced by permuting a generated domain name that resembles legitimate names. Linguistic (e.g. number of vowels) and statistical properties (e.g. letter frequencies) of the initial malignant name are inherited by derived names.

In the following we analyze specific feature contributions using summary plots derived by experimenting with malignant XTI's, randomly subsampled from the aforementioned DGA schemes:

- For *DirCrypt*, Fig. 7.4 shows that *Reputation* and *Length* are the most important features (higher SHAP values) followed by *Words_Mean*, *Freq_X*, *Freq_Q* and *Max_Let_Seq*. As expected, high *Length* and *Max_Let_Seq* values favor the malicious class since the typically long names and absence of digits discern *DirCrypt* names from benign ones. On the contrary, high values of *Reputation* and *Words_Freq* favor legitimate name classifications since *DirCrypt* names contain less whitelisted N-grams and meaningful words. High feature values of *Words_Mean* may be inconclusive, whereas lower *Words_Mean* values point to malignant (DGA) name categorizations.
- Regarding *Matsnu*, Fig. 7.5 shows that the most important features in terms of SHAP values are *Reputation*, *Words_Mean*, *Length*, *Vowel_Freq*, *Freq_B* and *Max_Let_Seq*. *Reputation* exclusively contributes to benign name classifications (negative SHAP values) since

many whitelisted N-grams may be present in both legitimate and *Matsnu* names, therefore favoring misclassifications (FN's) of DGA XTI's. High *Words_Mean* values point to benign name classifications (FN's); this is expected as *Matsnu* concatenates dictionary words that are typically short [203], thus higher *Words_Mean* values (mean length of meaningful words within the name) may mislead the classifier towards benign name classifications. *Reputation* and *Words_Mean* influence is mainly counterbalanced by *Length*, *Vowel_Freq* and *Freq_B* values. High *Length* values point to malicious name classifications since *Matsnu* names are typically longer than benign names. Although vowels are typically present in both benign and *Matsnu* names, high *Vowel_Freq* values enable DGA name categorizations (TP's); *Matsnu* names are usually longer than benign ones, hence they typically include more vowels. Letter *B* was found in various *Matsnu* XTI's, thus high *Freq_B* favors TP's.

- Regarding *Bamital*, Fig. 7.6 shows that *DeciDig_Freq*, *Length*, *Max_DeciDig_Seq* and *Reputation* mainly affect model decisions. High *DeciDig_Freq* values (i.e. total frequency of decimal digits 0-9) and high frequencies of specific hexadecimal digits (e.g. *Freq_C*, *Freq_D*, *Freq_B* and *Freq_1*) contribute significantly to TP's since *Bamital* names exclusively consist of such characters. As expected, impact of *Length* is very important since *Bamital* names follow MD5 hash function statistical properties and their size is fixed (i.e. 32 characters), thus clearly distinguishing them from benign names. High *Max_DeciDig_Seq* (i.e. maximum digit sequence) values point to misclassifications of DGA names as benign (FN's) since long decimal digit sequences are usually not present in *Bamital* names; hash function results are typically uniform, therefore short decimal digit sequences are followed by hexadecimal digits. High *Reputation* values erroneously favor the class of benign names (FN's) as the frequency of whitelisted N-grams within *Bamital* names is usually limited.
- For *VolatileCedar*, Fig. 7.7 shows that *Reputation* is the most important feature exclusively favoring legitimate classifications (FN's) with negative SHAP values; the initial name used by *VolatileCedar* resembles benign names, therefore many DGA N-grams may be included within the *Reputation whitelist*. The effect of *Reputation* is mainly counterbalanced by features *Length*, *Freq_E*, *Vowel_Freq*, *Freq_L* and *Max_Let_Seq*. As a permutation-based DGA, *VolatileCedar* is characterized by specific feature values, which act as signatures for discerning malicious names from benign ones.

7.5.4.3 XGBoost & MLP Classifier Dependence Plots for all DGA Families

In this subsection, SHAP dependence plots are used to investigate pairwise feature relationships, thus complementing our analysis based on summary plots. Fig. 7.8–7.15 depict XGBoost and MLP classifier dependence plots on malignant XTI's subsampled from all DGA families. Plots are provided for 4 features of interest, i.e. *Reputation*, *Entropy*, *Max_DeciDig_Seq* and *Words_Mean*. Interacting features are determined by SHAP using Shapley interaction val-

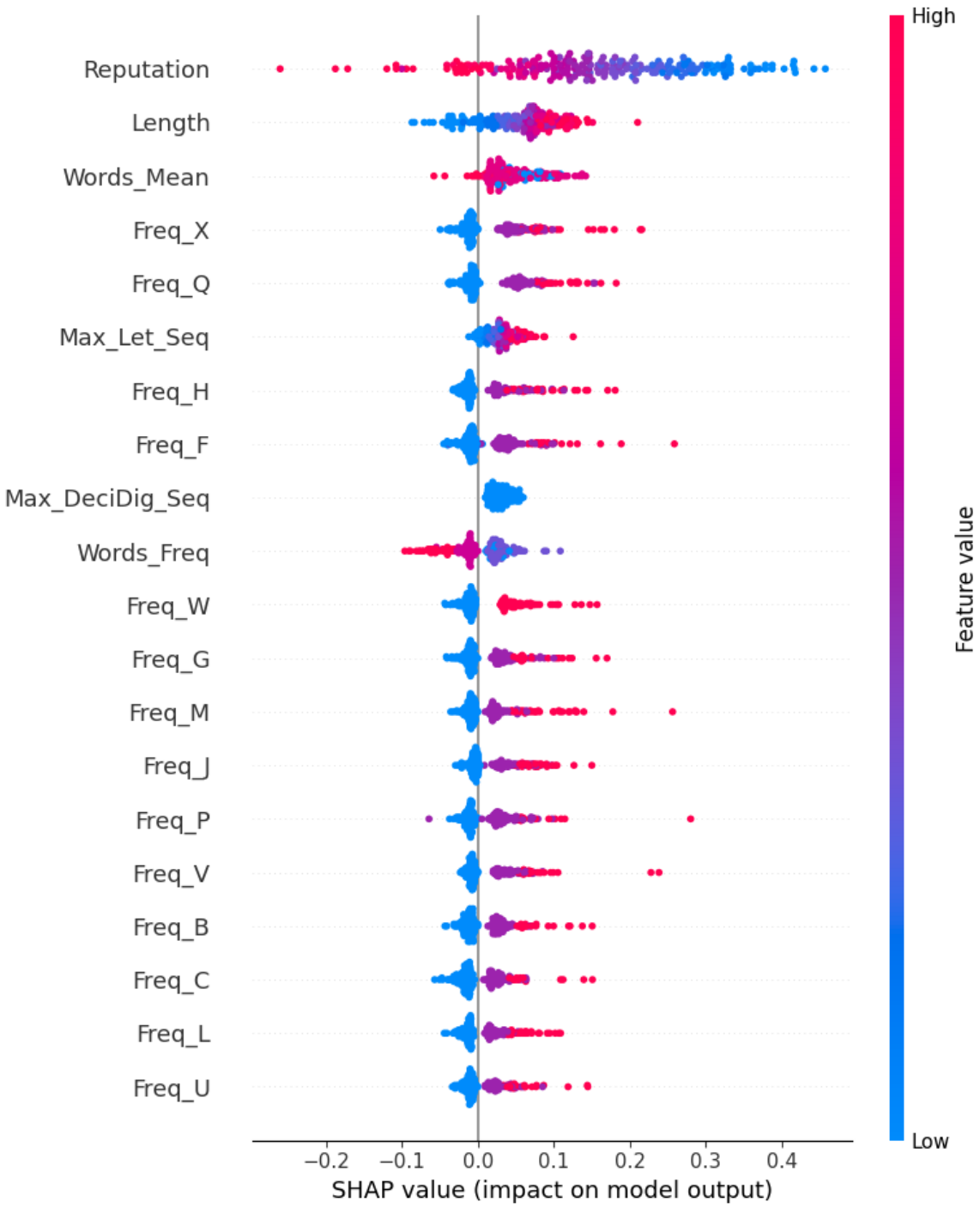

Figure 7.4: SHAP summary plot derived for XTI's from *DirCrypt* arithmetic-based DGA family (Algorithm: Multi-Layer Perceptron - MLP)

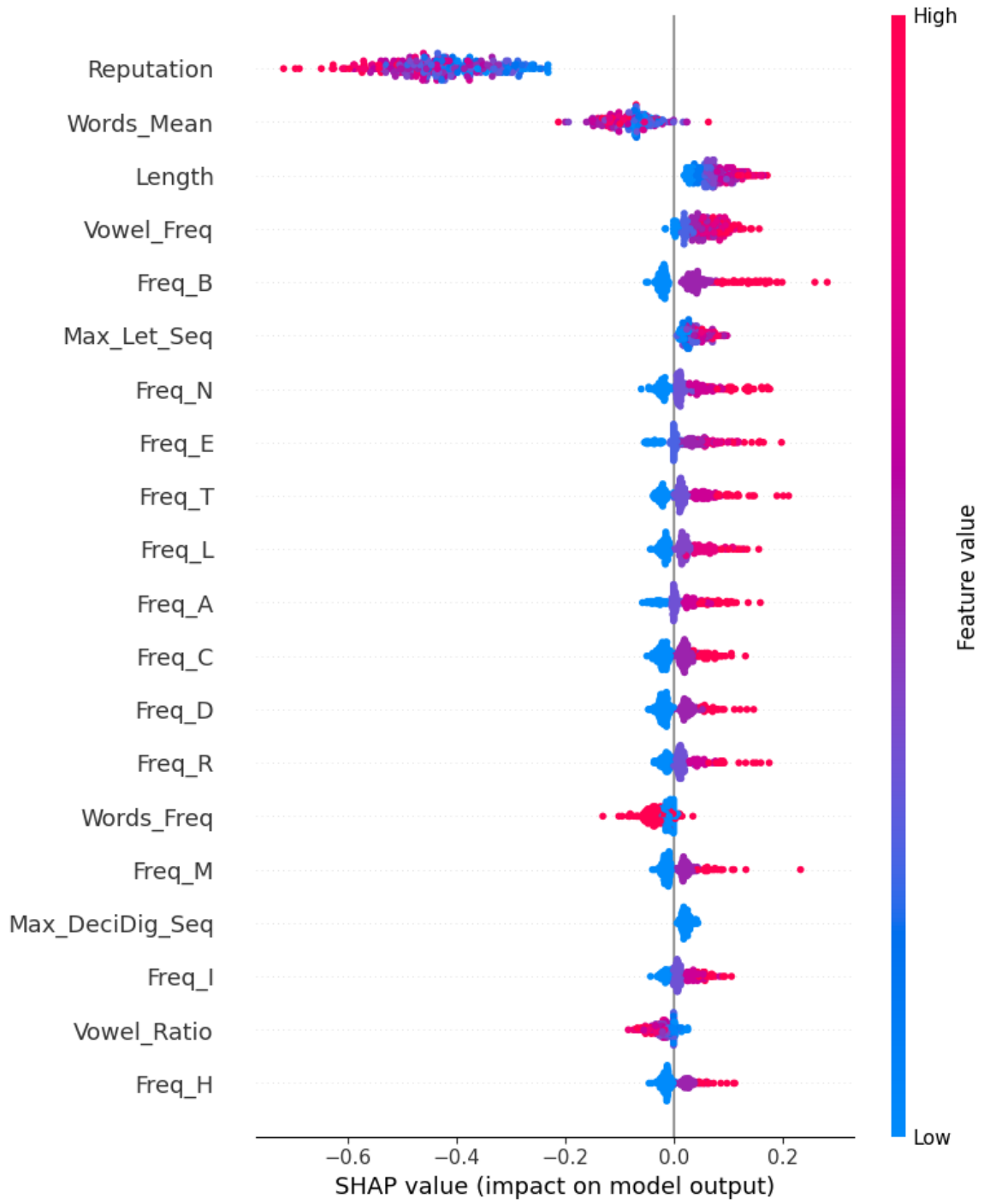

Figure 7.5: SHAP summary plot derived for XTI's from *Matsnu* wordlist-based DGA family (Algorithm: Multi-Layer Perceptron - MLP)

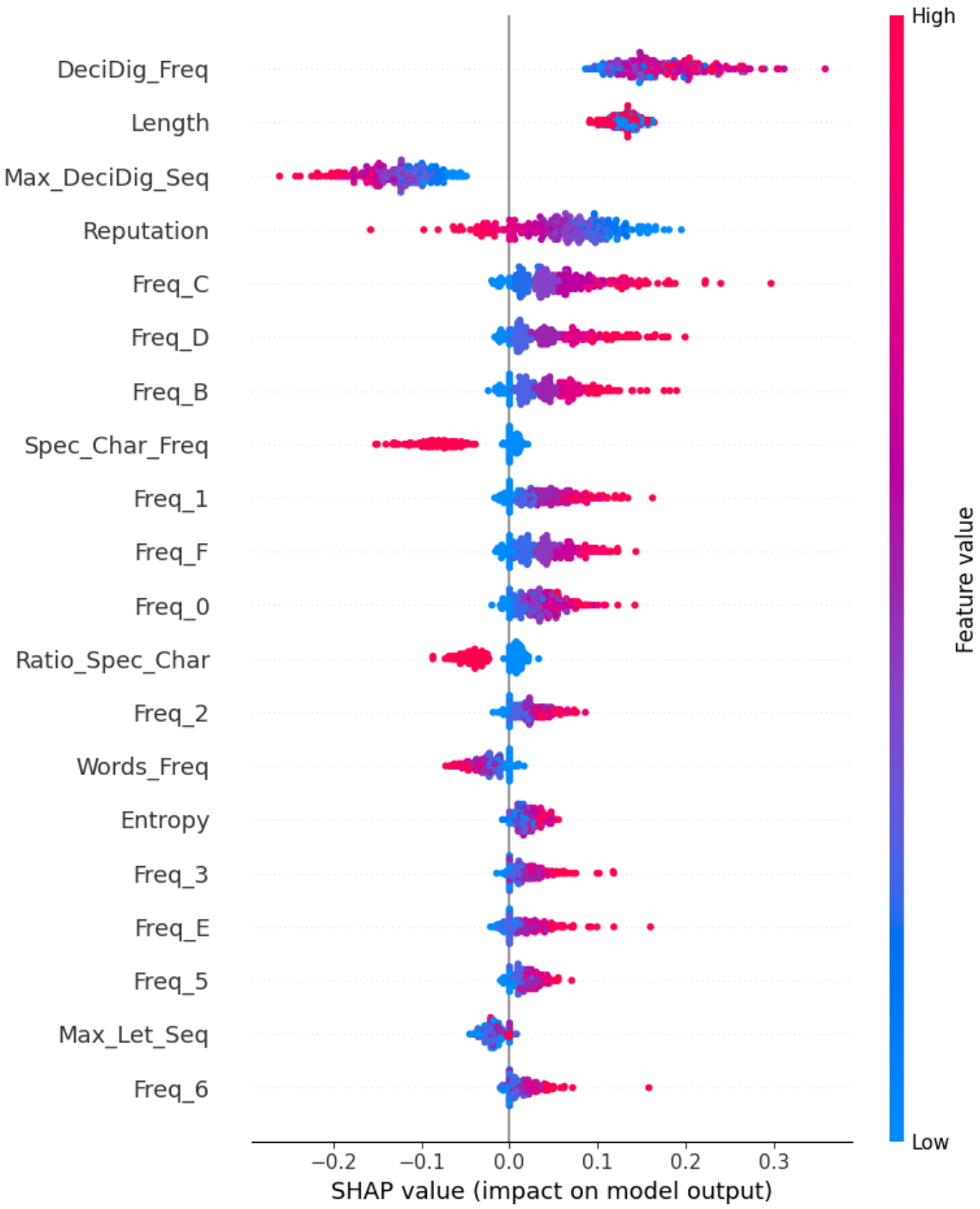

Figure 7.6: SHAP summary plot derived for XTI's from *Bamital* hash-based DGA family (Algorithm: Multi-Layer Perceptron - MLP)

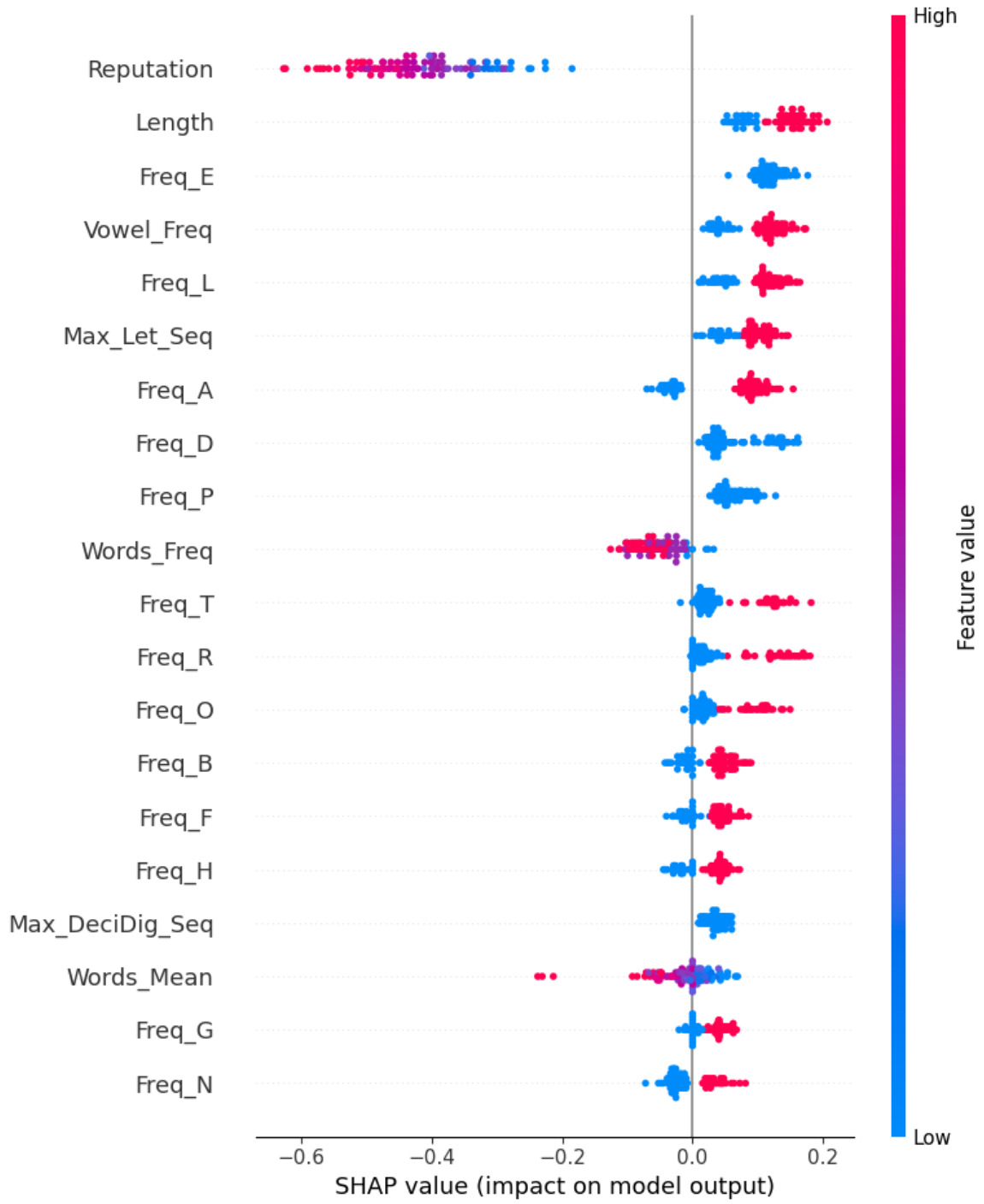

Figure 7.7: SHAP summary plot derived for XTI's from *VolatileCedar* permutation-based DGA family (Algorithm: Multi-Layer Perceptron - MLP)

ues (subsection 7.4.3); red and blue colors denote high and low values of interacting features respectively, while these values are depicted normalized between *0* and *1* (subsection 7.4.2). Interactions pertaining to features of interest are summarized below:

- ***Reputation Interactions*:** Fig. 7.8 and Fig. 7.9 show that *Reputation* significantly influences classifications. Namely, *Reputation* interacts with *Length* for XGBoost and *DeciDig_Freq* for MLP. However, combined *Reputation* and interacting feature values do not clearly affect classification decisions because, as shown in Fig. 7.2 and Fig. 7.3, the impact of *Reputation* is significantly higher than that of *Length* and *DeciDig_Freq*.
- ***Entropy Interactions*:** As expected from the summary plots of subsection 7.5.4.1, Fig. 7.10 and Fig. 7.11 show that *Entropy* values are not significant for both XGBoost and MLP classifiers. Although higher *Entropy* values may favor malicious name categorizations for MLP's, their SHAP values are considerably low, therefore *Entropy* effect is counterbalanced by more influential features.
- ***Max_DeciDig_Seq Interactions*:** As already mentioned in subsection 7.5.4.1, values of *Max_DeciDig_Seq* feature are not significant for XGBoost (Fig. 7.12). For MLP, Fig. 7.13 depicts that *Max_DeciDig_Seq* significantly impacts classifications and interacts with *Length*. Long sequences of decimal digits, i.e. high *Max_DeciDig_Seq* values, combined with shorter names, i.e. low *Length* values favor benign name classifications. This is expected as several DGA families alternate letters and decimal digits, thus long digit sequences are not formed.
- ***Words_Mean Interactions*:** Fig. 7.14 and Fig. 7.15 show that *Words_Mean* values affect both XGBoost and MLP classifiers. However, although high *Words_Mean* values favor legitimate name classifications (FN's) for MLP, for XGBoost high *Words_Mean* values mainly point to malicious name categorizations. Explicit correlations between *Words_Mean* and other interacting features are not evident in our experiments.

7.5.4.4 MLP Classifier Dependence Plots for Selected DGA Families

In this subsection we present indicative dependence plots for MLP's mapping eXplainability Test Instances (XTI's) for dominant features per DGA scheme (see subsection 7.5.4.2). Notably, in Fig. 7.16 – 7.19 we indicatively present dependence plots pertaining to *DirCrypt* and *Bamital*.

DirCrypt XTI's in Fig. 7.16 and Fig. 7.17 show that *Reputation* and *Length* features interact with *Max_Let_Seq* and *Reputation* respectively. High *Reputation* and *Max_Let_Seq* values favor benign class categorizations (FN's), while high *Length* values favor TP's. Such effect of *Reputation* and *Length* on model classifications is expected since *DirCrypt* names are typically long and randomized, thus they stand out from benign names. Note that *Length* effect on model decisions increases at a smaller rate as *Reputation* values increase.

Bamital XTI's in Fig. 7.18 and 7.19 show that *DeciDig_Freq* interacts with *Spec_Char_Freq*, while *Max_DeciDig_Seq* with *Length*. Increasing *DeciDig_Freq* favors TP's, with its influence

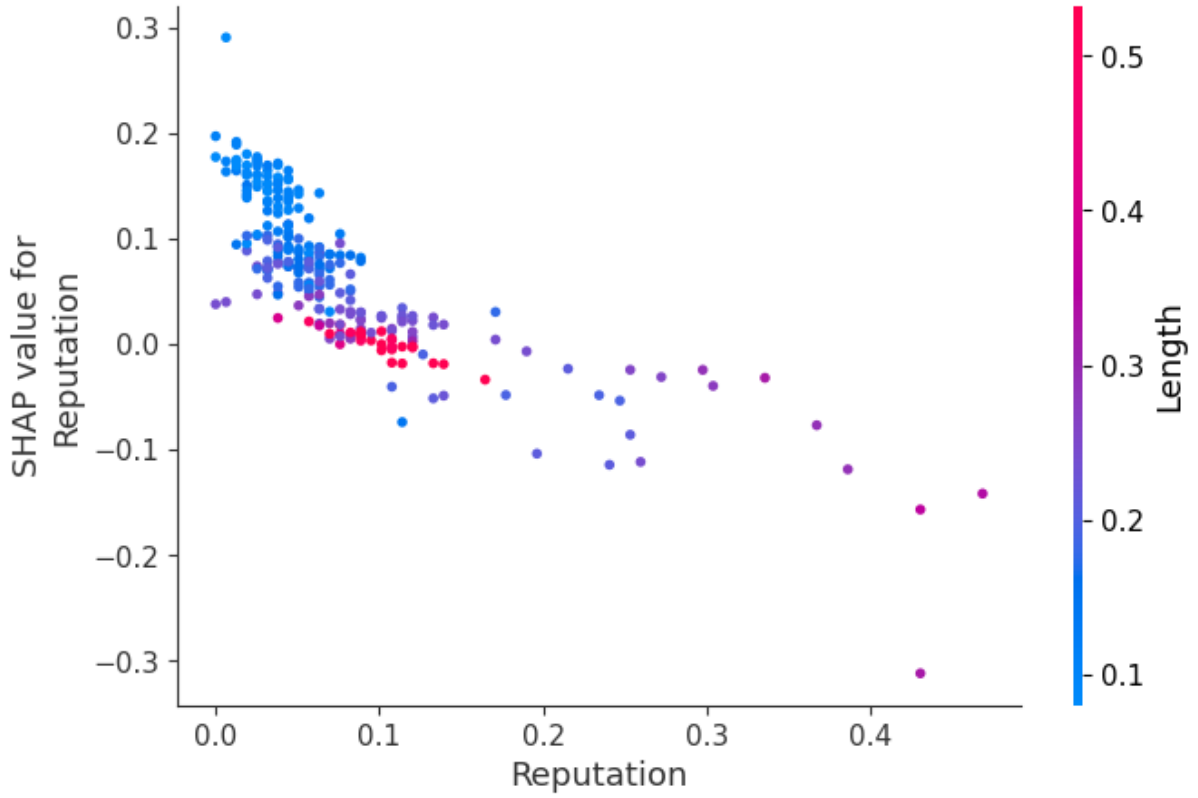

Figure 7.8: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Reputation*, Model: XGBoost

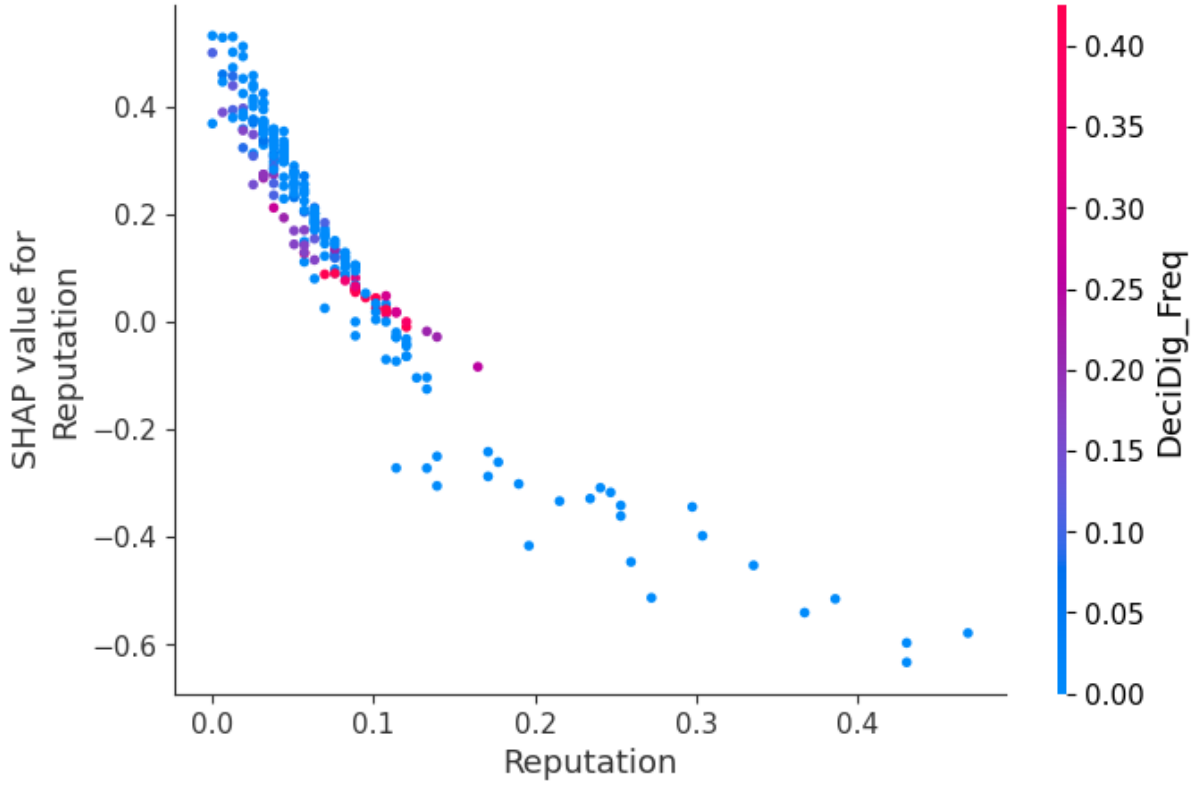

Figure 7.9: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Reputation*, Model: MLP

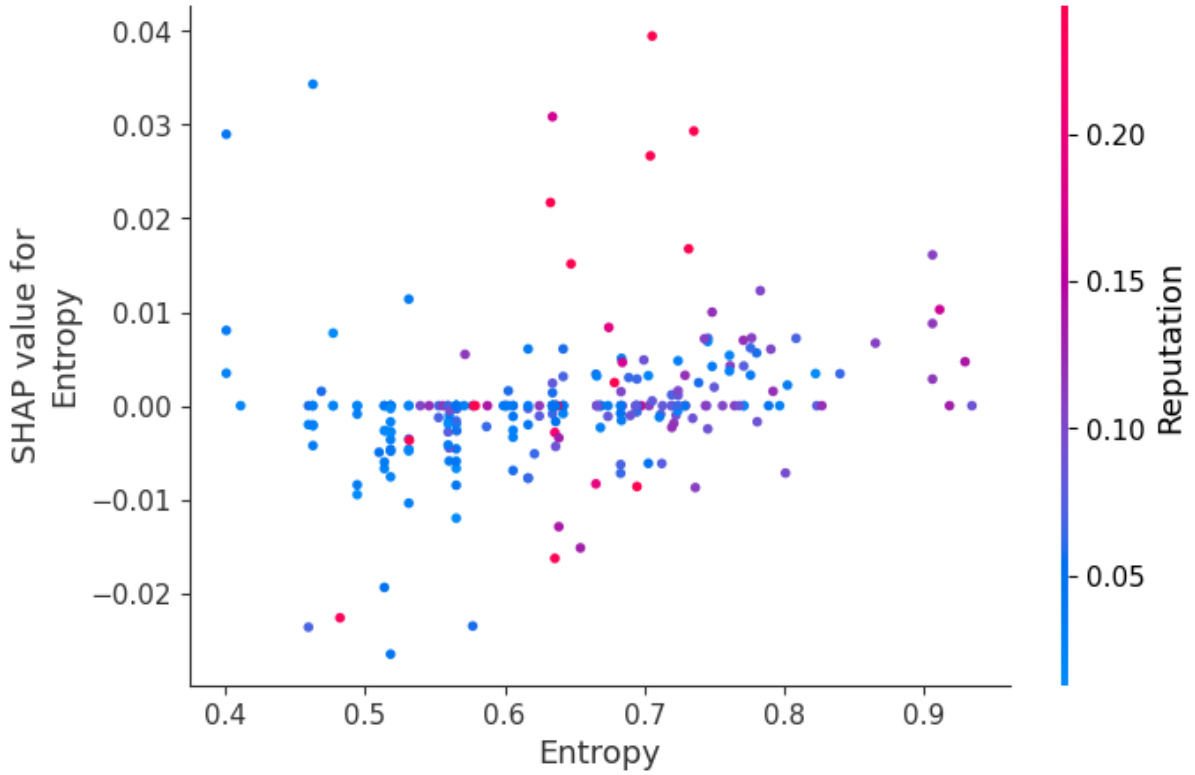

Figure 7.10: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Entropy*, Model: XGBoost

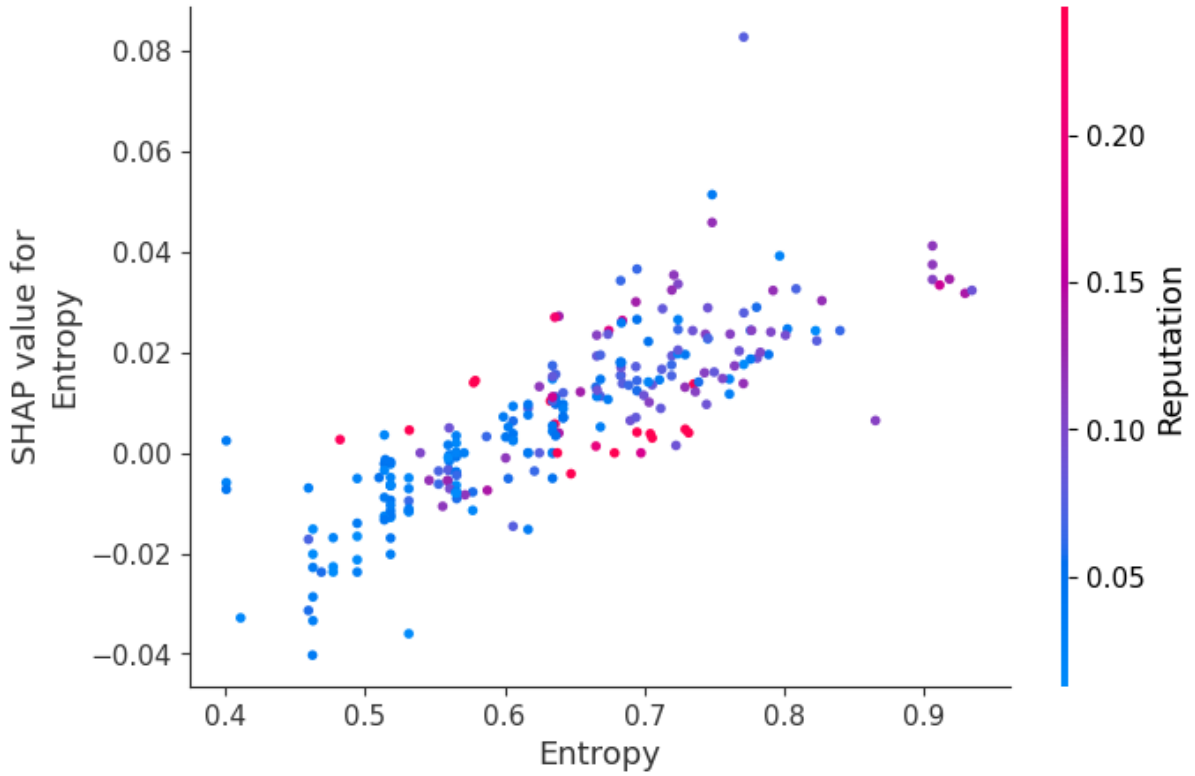

Figure 7.11: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Entropy*, Model: MLP

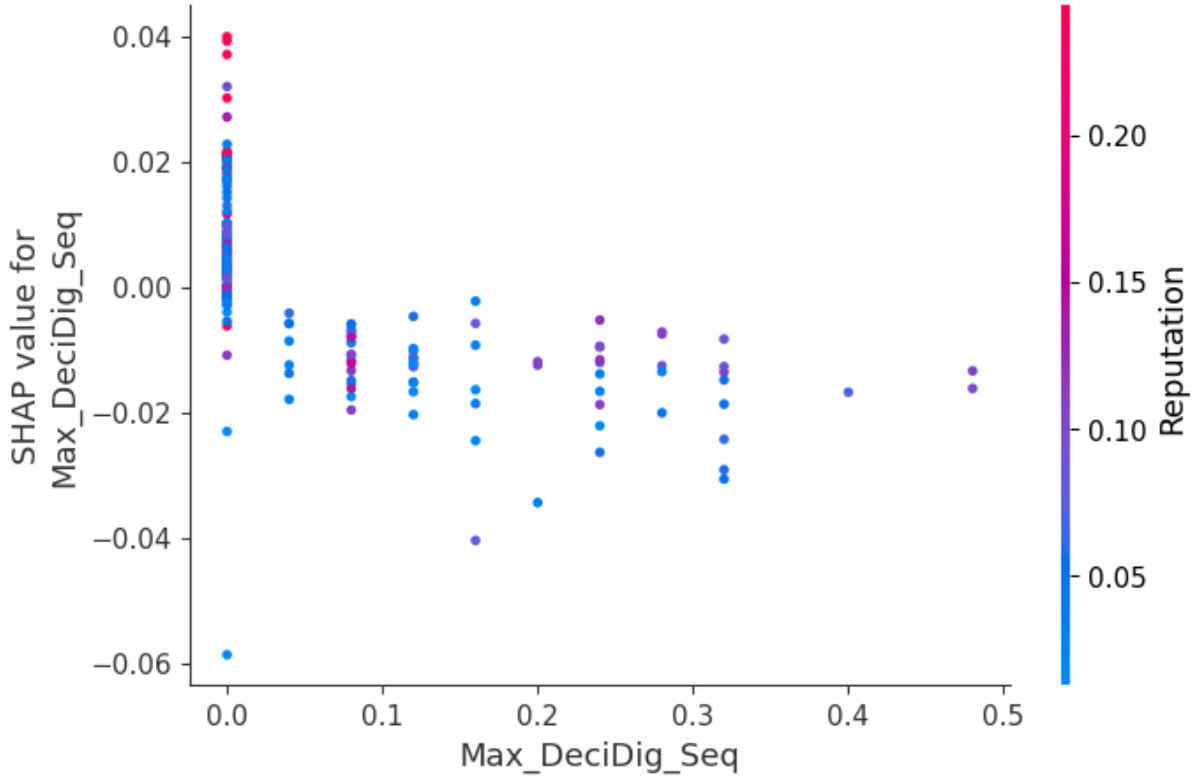

Figure 7.12: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Max_DeciDig_Seq*, Model: XGBoost

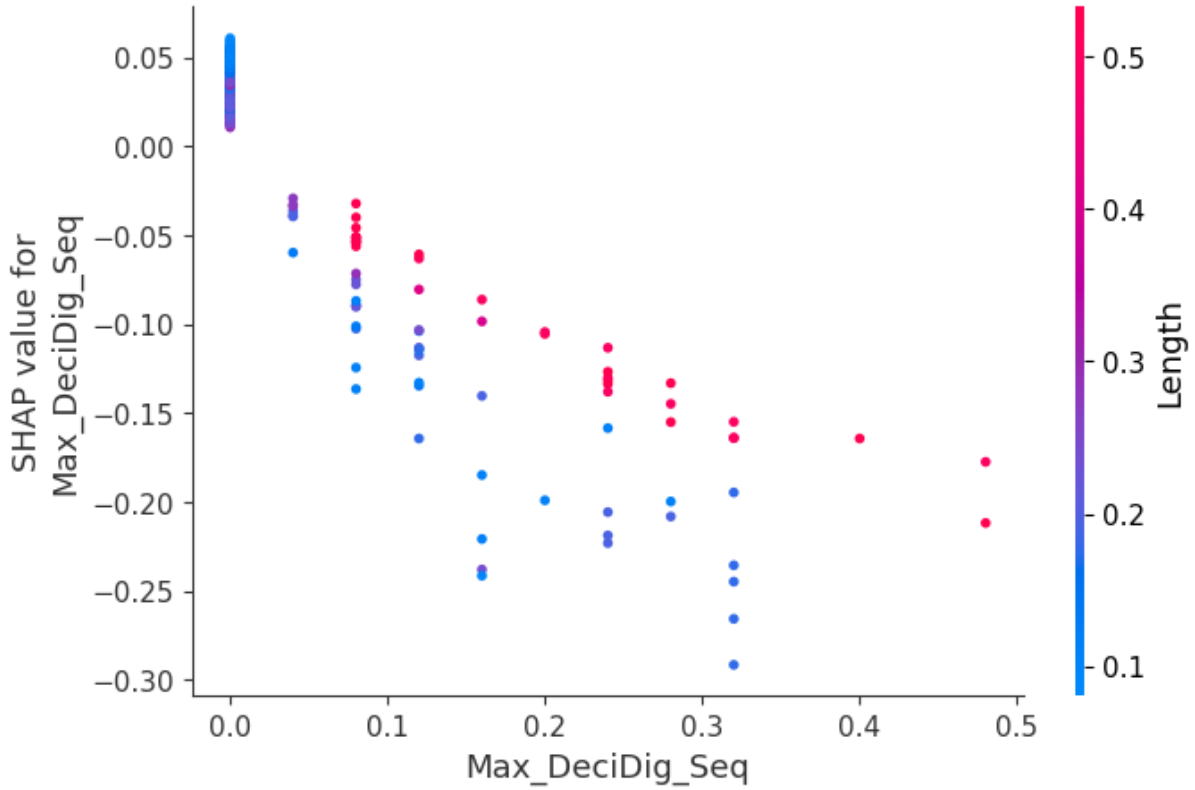

Figure 7.13: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Max_DeciDig_Seq*, Model: MLP

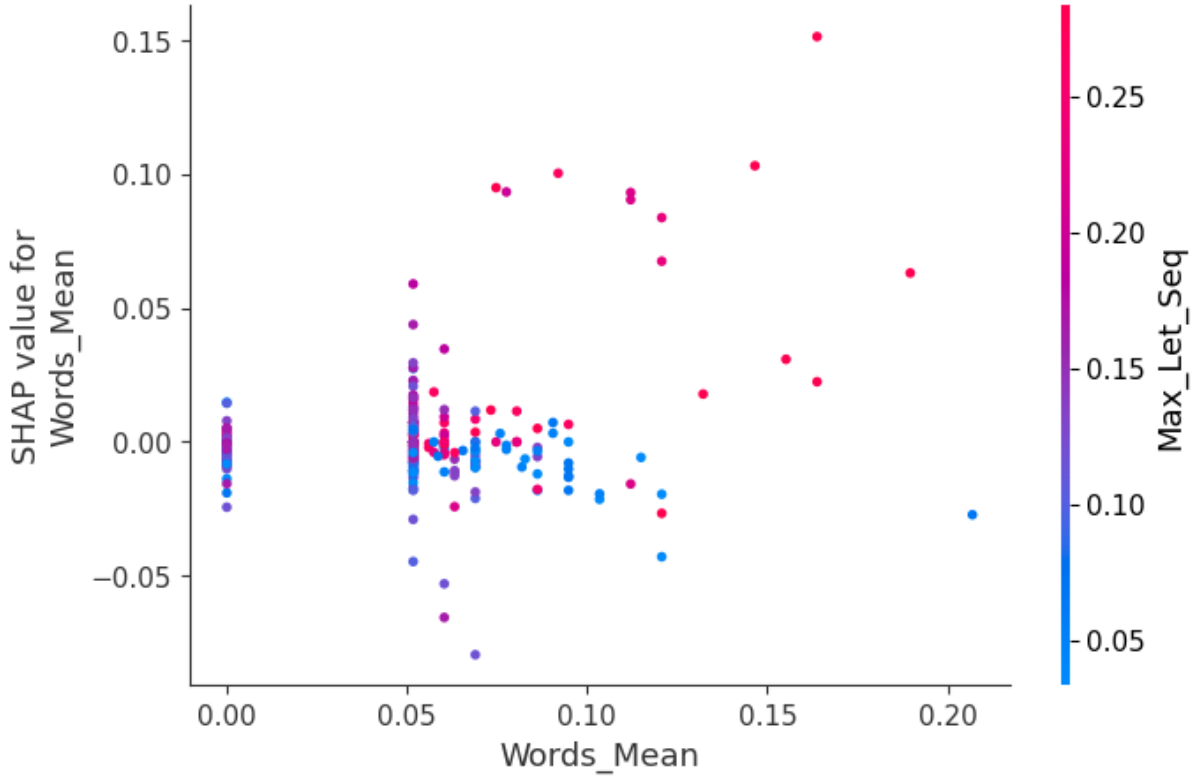

Figure 7.14: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Words_Mean*, Model: XGBoost

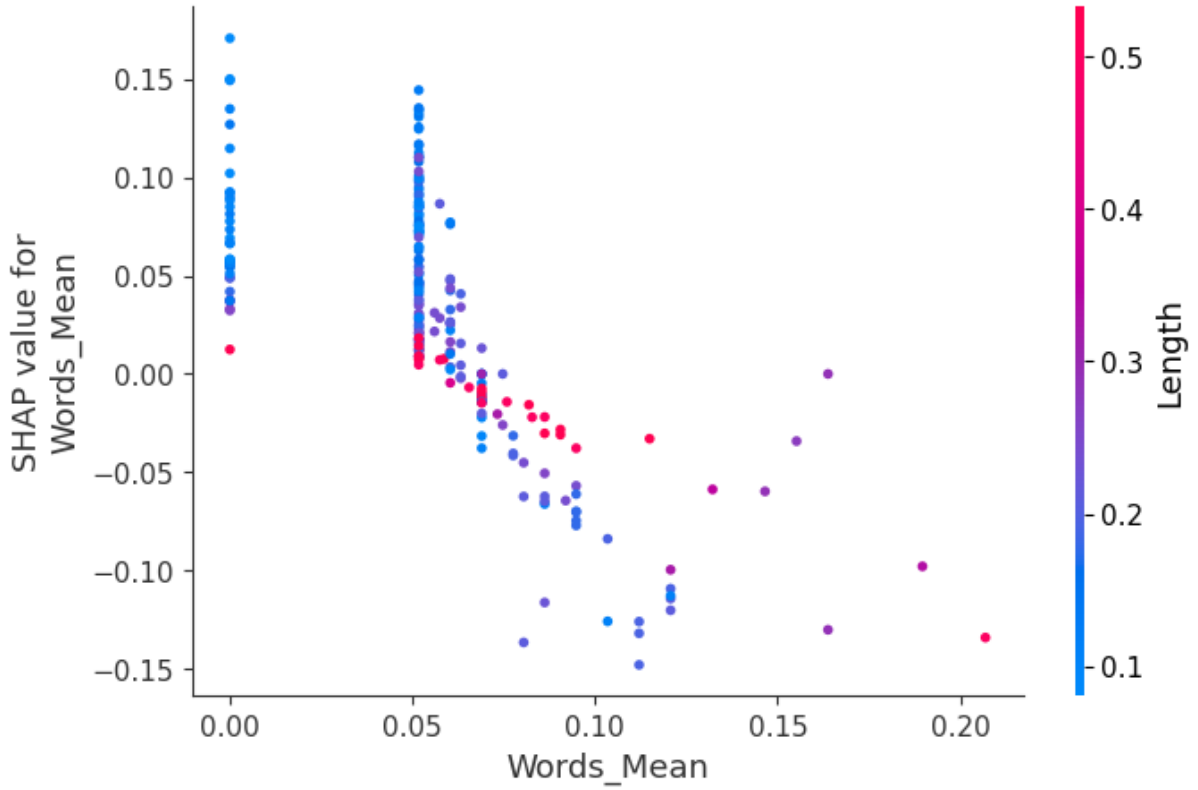

Figure 7.15: SHAP dependence plot derived for XTI's including malicious DNS names from all DGA families - Feature: *Words_Mean*, Model: MLP

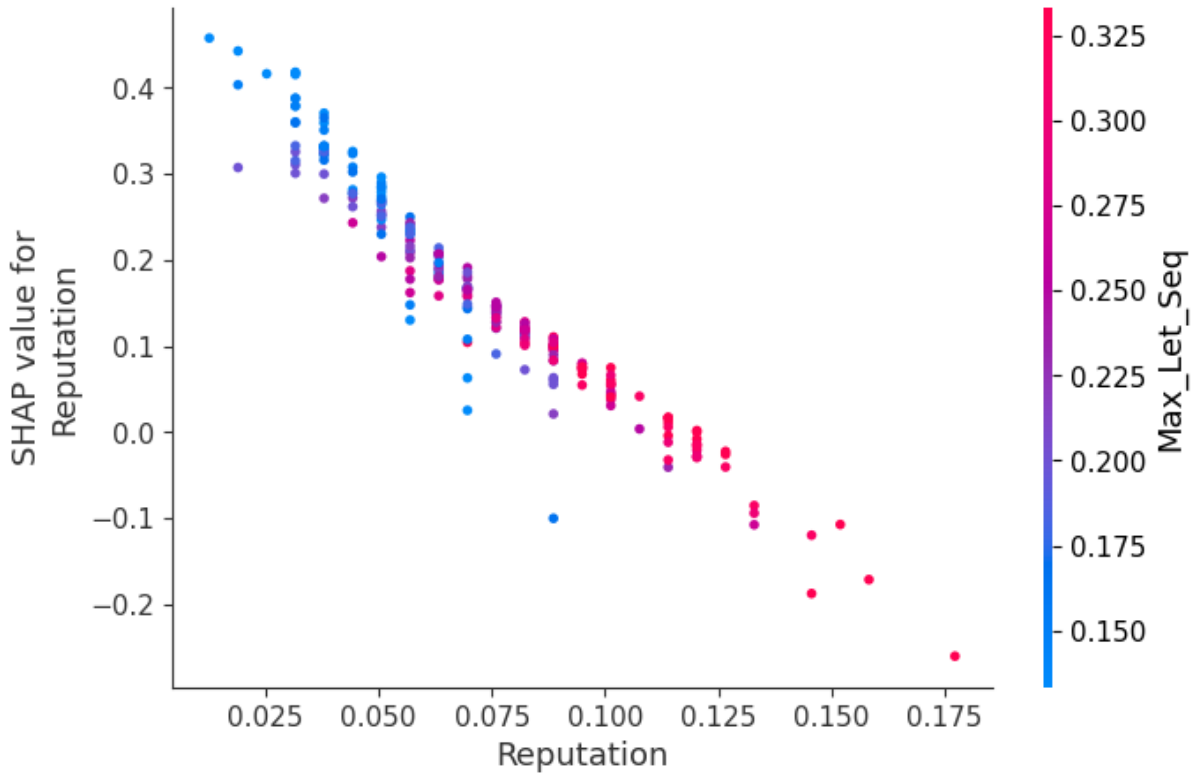

Figure 7.16: SHAP dependence plot derived for XTI's from *DirCrypt* DGA family - Feature: *Reputation* (Algorithm: Multi-Layer Perceptron - MLP)

increasing (higher SHAP values) for higher values of the interacting feature (*Spec_Char_Freq*). This is expected because *Bamital* names consist of hexadecimal digits, thus decimal digits constitute their majority. Moreover, as in Fig. 7.19, increased *Max_DeciDig_Seq* values favor FN's since *Bamital* follows the statistical properties of MD5 hash function with hexadecimal digits uniformly distributed across domain names. Therefore, long decimal digit sequences typically favor benign name misclassifications.

7.5.4.5 MLP Classifier Force Plots for Local Explainability

In this subsection force plots are used to analyze the operation of binary MLP classifiers pertaining to specific inputs (local explainability). Force plots are particularly helpful for understanding False Positives (FP's) and False Negatives (FN's) in classification of specific benign and DGA names. In these plots, features dominantly influencing name classifications are depicted along with their values. Red color denotes features favoring malicious name categorizations and blue colors those contributing to benign name classifications. A bold decimal value corresponds to the classifier output.

Fig. 7.20 and 7.21 depict force plots pertaining to MLP FP's, i.e. benign (non-DGA) names incorrectly categorized as DGA. Fig. 7.20 shows that name "wawibox.de" is perceived as DGA, mainly because of the high frequency of letter *W* and the low *Reputation* value. For this particular name *Freq_W* and the absence of many whitelisted N-grams override the effect of *Length*

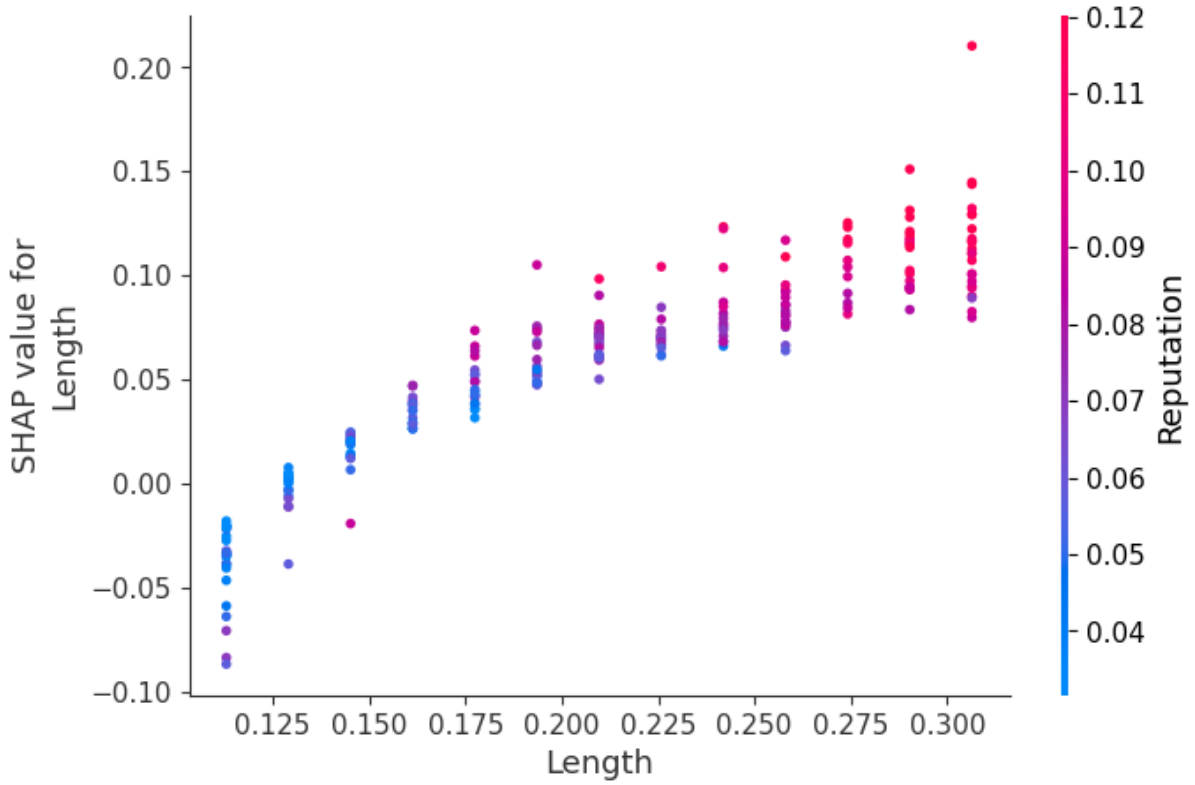

Figure 7.17: SHAP dependence plot derived for XTI's from *DirCrypt* DGA family - Feature: *Length* (Algorithm: Multi-Layer Perceptron - MLP)

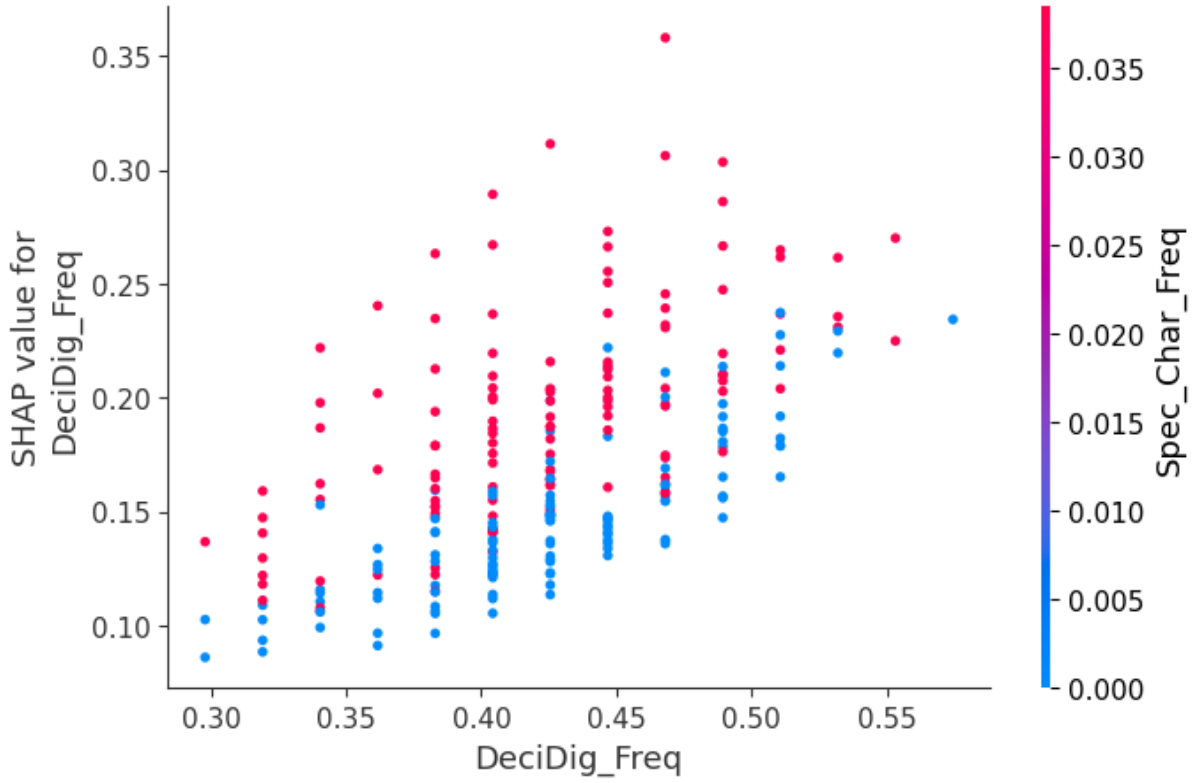

Figure 7.18: SHAP dependence plot derived for XTI's from *Bamital* DGA family - Feature: *DeciDig_Freq* (Algorithm: Multi-Layer Perceptron - MLP)

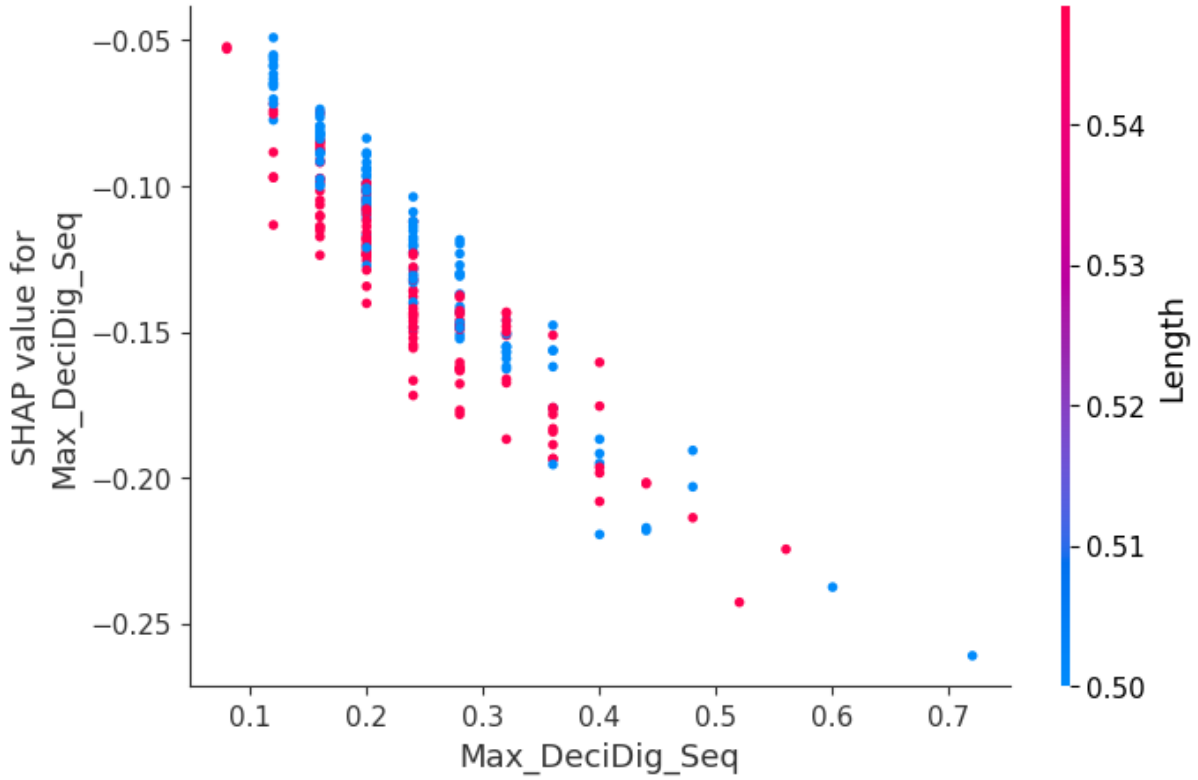

Figure 7.19: SHAP dependence plot derived for XTI's from *Bamital* DGA family - Feature: *Max_DeciDig_Seq* (Algorithm: Multi-Layer Perceptron - MLP)

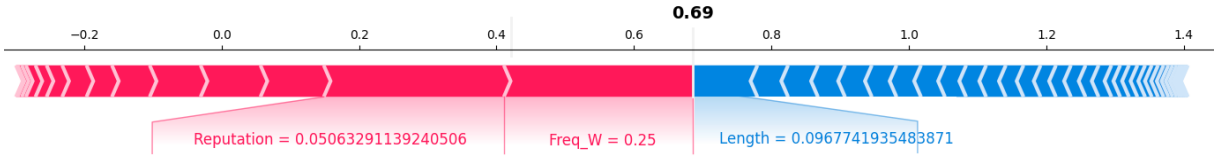

Figure 7.20: Force plot pertaining to benign (non-DGA) names incorrectly classified as DGA - Name: wawibox.de

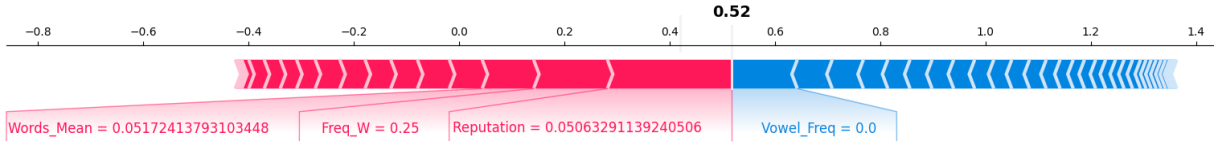

Figure 7.21: Force plot pertaining to benign (non-DGA) names incorrectly classified as DGA - Name: rvwgm2wrld2.xyz

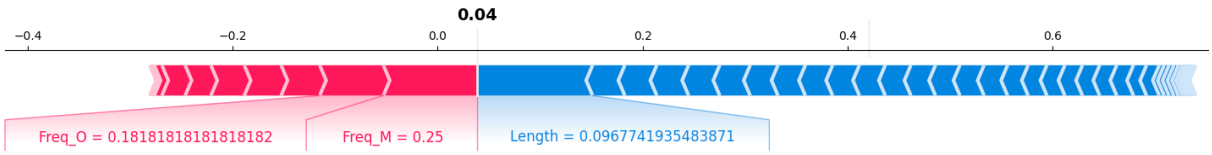

Figure 7.22: Force plot pertaining to malicious names incorrectly classified as benign - Name: nomodum.info, DGA Family: *Simda*

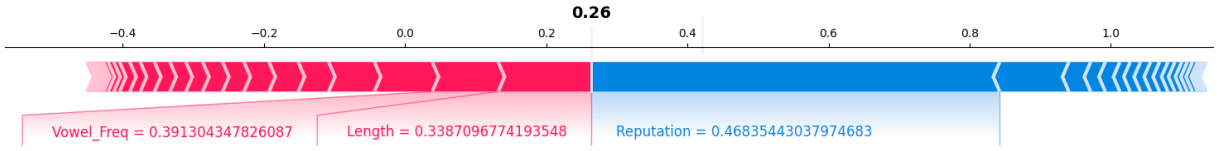

Figure 7.23: Force plot pertaining to malicious names incorrectly classified as benign - Name: californiatransferable.ru, DGA Family: *Gozi*

that favors benign name classifications. Fig. 7.21 shows that name "rvwgm2wrld2.xyz", which is frequently used for malware propagation [204] but is not produced by DGA's, is misclassified as DGA. This is attributed to the low *Reputation* value, the high frequency of letter *W* and the low *Words_Mean* value, although the zero *Vowel_Freq* value might point to non-DGA name classification.

Fig. 7.22 and 7.23 depict force plots pertaining to MLP FN's, i.e. DGA names incorrectly classified as benign. Fig. 7.22 shows that *Length* values have a major effect on misclassifying name "nomodum.info", generated by the *Simda* arithmetic-based DGA family, despite the high frequencies of letters *M* and *O* that favor malicious name classifications. Fig. 7.23 shows that name "californiatransferable.ru" originating from the *Gozi* wordlist-based DGA family is classified as benign because *Reputation* values point to benign name classifications. This counterbalances the effect of name length and the high presence of vowels that point towards DGA names.

7.6 Summary & Concluding Remarks

We investigated XAI methods for interpreting DGA name classifiers that detect malicious DNS messages used by bots to communicate with C&C servers. We addressed defense mechanisms based on ML classifiers and analyzed their operation via the SHAP algorithm that provides global and local interpretations in a model-agnostic, post-hoc manner.

To that end, we first configured tree-based and deep neural network binary classifiers for differentiating between benign DNS names and malicious names produced by DGA's. We trained and evaluated classifiers based on supervised ML algorithms, specifically RF's, GB, XGBoost, AdaBoost, ExtraTrees and MLP's. These relied on features directly extracted from domain name datasets, thus eliminating time-consuming and privacy-sensitive operations on repositories of historical data. Classifiers were trained using up-to-date and inclusive datasets. Legitimate names originated from Tranco, an online service ranking top Internet sites; we selected the 1 million most popular names. Malicious instances were sampled from the DGArchive repository, which reports 105 DGA families from 4 different generation schemes; we randomly selected 600,775 DGA names.

Our SHAP-based evaluation analyzed the features used by our trained XGBoost (determined as the most accurate tree-based model) and MLP deep neural network classifiers to segregate benign and DGA name instances. We investigated how DGA families and their different underlying algorithmic generation schemes (i.e. arithmetic, wordlist, hash or permutation based)

affect the features that specifically influence classification decisions. Relying on multiple SHAP visualization tools (summary, dependence and force plots) we provided global and local interpretations on sampled dataset instances. Specifically, we ranked feature importance, investigated the effect of feature values on model decisions and determined their interactions. Using up-to-date and extensive datasets, we conclude that our SHAP-based analysis enables interpretations of XGBoost and MLP name classifiers, attacked by well-known diverse DGA schemes. Such methods may facilitate ML adoption within networking environments where interpretations for black-box schemes are required.

Chapter 8

Conclusions & Future Steps

In this chapter we summarize our work (Section 8.1) and discuss future steps (Section 8.2).

8.1 Summary & Concluding Remarks

Motivated by the vital role of Domain Name System (DNS) for critical Internet services, we proposed mechanisms for effectively defending against major DNS cybersecurity threats. We considered DDoS attacks targeting DNS infrastructures (i.e. Water Torture) as well as techniques abusing DNS to establish botnet communications, i.e. Domain Generation Algorithms (DGA's). Our developed systems relied on various widely applied methods originating from diverse research fields, such as Big Data analytics and Software-Defined Networking. Specifically, we leveraged on probabilistic data structures, Machine Learning (ML) algorithms, eXplainable Artificial Intelligence (XAI) techniques and Programmable Data Planes (PDP's) to efficiently counter DNS cyber-attacks.

We initially proposed a user-space schema for the detection and mitigation of DNS Water Torture attacks targeting Authoritative DNS Servers (Chapter 3). Our approach relied on data structures and algorithms, which were capable of differentiating between legitimate and malicious DNS messages in a time and space efficient manner. We mainly emphasized on the utilization of Bloom Filters (BF's) for accurately mapping large DNS zones and rapidly filtering ingress DNS traffic. We also outlined the potential of supporting schemas for collaborative Water Torture attack mitigation driven by the privacy properties of BF's. Our analysis concluded that our proposed schema was able to effectively safeguard Authoritative DNS Servers without requiring extensive computational resources.

We subsequently extended our previously proposed user-space filtering mechanism (Chapter 3) by employing data plane programmability techniques for high-speed Water Torture mitigation at the NIC driver level of Authoritative DNS Servers. In Chapter 4, we employed eXpress Data Path (XDP) for rapid Deep Packet Inspection (DPI) of ingress DNS messages at the earliest level of the Linux kernel, i.e. before any memory is allocated to them. Therefore, our solution enabled efficient BF-based domain name lookups for segregating valid from invalid DNS requests entirely within the data plane of Authoritative DNS Servers. Contrary to other

data plane programmability methods (e.g. P4 switches and DPDK), our XDP-based solution does not depend on specialized hardware and the Linux kernel is not bypassed, hence network administrators may benefit from COTS hardware (e.g. low-cost NIC's) and the available Linux kernel modules (e.g. TCP/IP libraries). Our experimentation concluded that our proposed BF-based data plane mitigation mechanism was capable of significantly accelerating name lookups compared to our previously proposed user-space mechanism.

In Chapter 5, we proposed a privacy-aware schema to facilitate Authoritative DNS Server zone exchanges with no requirements for formal collaboration agreements. Via our schema, domain names may be distributed to external filtering mechanisms, e.g. Recursors or upstream cloud scrubbing infrastructures, which will mitigate DNS-based DDoS attacks more effectively closer to their origins or filter DGA traffic more accurately. Our mechanism relied on Cuckoo Filters (CF's) to map Authoritative DNS Server zones, which were selected instead of BF's due to their time, space and dynamic deletion advantages. Our experimental analysis concluded that our CF-based mechanism is promising for the privacy-aware distribution of Authoritative DNS Server zone contents.

Although the results of our whitelist-based DNS attack mitigation mechanisms were promising, their application is not possible when Authoritative DNS Server administrators are not willing to collaborate and thus, zone contents are not available to third-party mitigation systems. Therefore, in Chapter 6, we complemented our whitelist-based solutions by implementing anomaly-based filtering mechanisms for Water Torture attacks and/or DGA traffic. Specifically, we leveraged on XDP to accelerate ML classification within the data plane of Recursors. We developed Naive Bayes Classifiers, which were capable of effectively differentiating between benign and malicious DNS requests entirely within the Linux kernel of filtering appliances. Thus, the cost of ML inference at the name server control plane was avoided. When deployed within Recursors, our analysis demonstrated that our XDP-based solution accomplished significant mitigation throughput without requiring any specialized hardware, e.g. Graphics Processing Units (GPU's), for fast ML classification.

Finally, in Chapter 7 we relied on eXplainable Artificial Intelligence (XAI) techniques to interpret the operation of DGA name classifiers that differentiate between legitimate and malicious domain names. Specifically, we employed SHapley Additive exPlanation (SHAP) and various of its available visualization tools (summary, dependence, force plots) to assess feature importance and determine interactions among the selected features. As a model-agnostic and post-hoc XAI method, SHAP enabled the analysis of tree (i.e. XGBoost) and deep neural network (i.e. MLP) classifiers in a unified manner after the completion of the ML model learning phase. Features were extracted directly from domain names, thus they reported the statistical and linguistic properties of dataset entries, while costly and privacy sensitive external repository operations were avoided. Contrary to related approaches, our evaluation focused on analyzing classification decisions with respect to well-known fundamental DGA schemes (i.e. arithmetic, wordlist, hash and permutation based), which are used to construct diverse DGA names. Our evaluation concluded that SHAP is promising for understanding the functionality of ML-based

security mechanisms and thus, XAI methods may facilitate ML adoption within networking environments where justifications of black-box model decisions are required.

8.2 Recommendations for Further Research

In this section, we outline research directions, which may be considered to extend our proposed DNS cyber-attack protection solutions and provide more advanced security services.

Distributed Ledger Technology (DLT), e.g. Blockchain-based architectures [205, 206], may be investigated to streamline collaboration between Authoritative DNS Servers and potential attack mitigators (e.g. Recursors) via Smart Contracts [205]. Similarly to [207], permissioned Blockchains may be considered because of their immutability properties to ensure transparent transactions among collaborators. Collaborator mitigation requests may also be ranked based on reputation-based systems, whereas appropriate mechanisms may be developed to provide incentives for mitigation offers.

Recently proposed probabilistic data structures, e.g. Morton Filters [208], Xor Filters [209] and/or Vacuum Filters [210], may be applied to improve our BF and CF based mechanisms for mapping DNS zones. These have been reported to outperform BF's and CF's in terms of lookup latency and memory requirements, while permitting flexible element updates. The aforementioned data structures could be employed for privacy-aware zone exchanges and they may be implemented within the data plane using the XDP framework to assess if the mitigation throughput is improved compared to our proposed BF-based filtering solution.

Apart from supporting Water Torture detection and mitigation, our proposed mechanisms may also be adapted to additional DNS attack vectors, such as DNS Flood, DNS Amplification, NXNSAttacks [165] and TsuNAME DNS attacks [211]. Comparisons with existing countermeasures, e.g. [212], in terms of latency and memory requirements could also be considered. Additionally, potential extensions of our proposed mechanisms may be investigated to defend against malicious attempts that are based on encrypted DNS traffic [213].

Our XDP-based data plane mechanisms may be compared with other data plane programmability technologies, e.g. P4 switches and/or DPDK. Furthermore, XDP and eBPF could be used to implement data plane ML inference for additional algorithms, e.g. Multi-Layer Perceptrons (MLP's) and/or Random Forests (RF's); their throughput and accuracy may be compared with our proposed in-network Naive Bayes Classifiers. Moreover, alternative techniques for mapping ML algorithm parameters within the data plane, e.g. those reported in [83], may accomplish higher DDoS attack mitigation throughput than directly implementing ML algorithm mathematical equations.

Our SHAP-based interpretations may be extended to address additional deep neural network models. These may include Convolutional Neural Networks (CNN's), Long Short-Term Memory (LSTM) networks and/or Bidirectional LSTM (BiLSTM) networks, which can be employed for DGA name classification [60]. Alternative XAI approaches, e.g. LIME [176] and Counterfactual Explanation [177], may also be considered. The proposed scheme may be further adapted

to unsupervised deep learning models, e.g. Autoencoders.

Finally, the proposed mechanisms may be extended to multi-domain infrastructures using Federated Learning [214] for collaborative DGA name detection, similarly to [215,216]. Therefore, collaborators may converge to federated ML models without sharing any private attack and benign data.

Chapter 9

Publications

This chapter lists our articles in scientific journals (Section 9.1) and our papers in peer-reviewed conferences and workshops (Section 9.2).

9.1 Articles in Scientific Journals

- N. Kostopoulos, D. Kalogeras, D. Pantazatos, M. Grammatikou, and V. Maglaris, *SHAP Interpretations of Tree and Neural Network DNS Classifiers for Analyzing DGA Family Characteristics*, in IEEE Access, Volume 11, pp. 61144 – 61160, June 2023
- A. Pavlidis, M. Dimolianis, K. Giotis, L. Anagnostou, N. Kostopoulos, T. Tsigkritis, I. Kotinas, D. Kalogeras, and V. Maglaris, *Orchestrating DDoS Mitigation via Blockchain-based Network Provider Collaborations*, in The Knowledge Engineering Review, Volume 35, e16, pp. 1–17, April 2020

9.2 Papers in Peer-reviewed Conferences & Workshops

- M. Dimolianis, D. Kalogeras, N. Kostopoulos, and V. Maglaris, *DDoS Attack Detection via Privacy-aware Federated Learning and Collaborative Mitigation in Multi-domain Cyber Infrastructures*, in the Proceedings of the 11th IEEE International Conference on Cloud Networking (IEEE CloudNet 2022), Paris, France, November 2022, pp. 118–125
- N. Kostopoulos, P. V. Vuletić, K. Stamos, E. Gaci, P. Hita, D. Bixheku, N. Gamtsemlidze, T. Gozalishvili, and K. Baumann, *Leveraging on WiFiMon for Efficient Wi-Fi Performance Monitoring (Demo)*, in the Proceedings of the 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob 2022), Thessaloniki, Greece, October 2022
- N. Kostopoulos, S. Korentis, D. Kalogeras, and V. Maglaris, *Mitigation of DNS Water Torture Attacks within the Data Plane via XDP-Based Naive Bayes Classifiers*, in the Proceedings of the 10th IEEE International Conference on Cloud Networking (IEEE CloudNet 2021), Virtual Event, November 2021, pp. 133–139

- N. Kostopoulos, S. Gjeçi, K. Baumann, P. V. Vuletić, and K. Stamos, *WiFiMon: Combining Crowdsourced and Probe Measurements for Wi-Fi Performance Evaluation*, in the Proceedings of the 16th IEEE/IFIP Annual Conference on Wireless On-demand Network Systems and Services (IEEE/IFIP WONS 2021), Virtual Event, March 2021, pp. 1–8
- N. Kostopoulos, D. Kalogeras, and V. Maglaris, *Enabling Privacy-Aware Zone Exchanges Among Authoritative and Recursive DNS Servers*, in the Proceedings of the ACM/IRTF Applied Networking Research Workshop 2020 (ACM/IRTF ANRW'20), Virtual Event, July 2020, pp. 1–8
- N. Kostopoulos, D. Kalogeras, and V. Maglaris, *Leveraging on the XDP Framework for the Efficient Mitigation of Water Torture Attacks within Authoritative DNS Servers*, in the Proceedings of the 6th IEEE International Conference on Network Softwarization (IEEE NetSoft 2020), Virtual Event, June 2020, pp. 287–291
- N. Kostopoulos, A. Pavlidis, M. Dimolianis, D. Kalogeras, and V. Maglaris, *A Privacy-Preserving Schema for the Detection and Collaborative Mitigation of DNS Water Torture Attacks in Cloud Infrastructures*, in the Proceedings of the 8th IEEE International Conference on Cloud Networking (IEEE CloudNet 2019), Coimbra, Portugal, November 2019, pp. 1–6
- K. Giotis, A. Pavlidis, L. Anagnostou, M. Dimolianis, T. Tsigkritis, D. Kalogeras, N. Kostopoulos, I. Kotinas, and V. Maglaris, *Blockchain-based Federation of Network Providers for Collaborative DDoS Mitigation*, in the Proceedings of the 3rd Symposium on Distributed Ledger Technology, Gold Coast, Australia, November 2018

Chapter 10

Extended Abstract in Greek - Εκτεταμένη Περίληψη στα Ελληνικά

Το κεφάλαιο αυτό περιλαμβάνει μία εκτεταμένη περίληψη της διατριβής στην ελληνική γλώσσα. Κάθε ενότητα της εκτεταμένης περίληψης αφορά ένα κεφάλαιο του αγγλικού κειμένου. Στο τέλος του κεφαλαίου περιλαμβάνεται γλωσσάριο στο οποίο παρουσιάζονται οι αντιστοιχίες ελληνικών και αγγλικών όρων.

10.1 Κεφάλαιο 1

Το Σύστημα Ονοματοδοσίας (Domain Name System, DNS) παρέχει μηχανισμούς για την απεικόνιση συμβολικών αναγνωριστικών (ονόματα – domain names) σε διάφορα είδη πληροφοριών. Χαρακτηριστικό παράδειγμα τέτοιων πληροφοριών αποτελούν οι αριθμητικές τιμές που αντιστοιχούν σε διευθύνσεις πρωτοκόλλου Διαδικτύου (Internet Protocol, IP). Οι περισσότερες δικτυακές υπηρεσίες βασίζονται σε domain names για την ομαλή λειτουργία τους. Κατά συνέπεια, η σημασία του DNS είναι ζωτική για το Internet.

Λόγω της σπουδαιότητάς του, το DNS γίνεται συχνά στόχος κυβερνοεπιθέσεων, όπως είναι οι κατανεμημένες επιθέσεις άρνησης παροχής υπηρεσίας (Distributed Denial of Service attacks ή επιθέσεις DDoS). Οι επιθέσεις DDoS πλημμυρίζουν τα θύματά τους με μεγάλο όγκο κίνησης για να κατασπαταλήσουν τους διαθέσιμους πόρους τους, π.χ. την επεξεργαστική ισχύ, τη διαθέσιμη μνήμη ή/και το εύρος ζώνης των ζεύξεων που οδηγούν στα θύματα.

Μία επίθεση DDoS εναντίον του DNS με πολύ σοβαρές συνέπειες είναι η Water Torture. Θύματα της επίθεσης είναι επίσημοι εξυπηρετητές DNS (Authoritative DNS Servers). Οι εξυπηρετητές αυτοί διατηρούν αντιστοιχίες ονομάτων και πληροφοριών DNS, ενώ είναι υπεύθυνοι για συγκεκριμένες ζώνες, δηλαδή υποσύνολα της υποδομής DNS που βρίσκονται υπό τη διαχείριση ενός συγκεκριμένου οργανισμού. Παράπλευρα θύματα της επίθεσης Water Torture είναι και οι αναδρομικοί εξυπηρετητές DNS (Recursive DNS Servers ή Recursors), που προωθούν την επιθετική κίνηση στους Authoritative DNS Servers. Οι Recursors είναι εξυπηρετητές DNS, οι οποίοι αναζητούν τους Authoritative DNS Servers που εξυπηρετούν τις ζώνες στις οποίες απευθύνονται τα ερωτήματα DNS των επιτιθέμενων.

Οι επιθέσεις Water Torture πλημμυρίζουν τους Authoritative DNS Servers μίας ζώνης με μεγάλο πλήθος ερωτημάτων DNS. Σκοπός της επίθεσης είναι η εξάντληση της υπολογιστικής ισχύος των θυμάτων. Κύριο χαρακτηριστικό της επίθεσης είναι ότι τα κακόβουλα ερωτήματα αφορούν domain names που δεν είναι αποθηκευμένα στις ζώνες του θύματος. Παράγοντας με τυχαίο τρόπο το πρόθεμα (prefix) των domain names εξασφαλίζεται ότι παρακάμπτονται οι προσωρινές μνήμες DNS (DNS cache) των Recursors. Κατά συνέπεια, όλη η επιθετική κίνηση κατευθύνεται στο θύμα, μεγιστοποιώντας τις συνέπειες της επίθεσης.

Εκτός από τις επιθέσεις DDoS, οι επιτιθέμενοι κακομεταχειρίζονται συχνά το DNS για την εγκατάσταση διαύλων επικοινωνίας στους οποίους προωθούνται κακόβουλα δεδομένα με χαμηλούς ρυθμούς, ώστε να αποφεύγεται ο εντοπισμός τους. Οι επιτιθέμενοι ουσιαστικά εκμεταλλεύονται τις πολιτικές τειχών προστασίας (firewalls) που εφαρμόζονται σε δίκτυα υπολογιστών. Οι πολιτικές αυτές δεν μπλοκάρουν τα μηνύματα DNS λόγω της αναγκαιότητάς τους για την ομαλή λειτουργία πολλών δικτυακών υπηρεσιών.

Χαρακτηριστικό παράδειγμα τέτοιων κακόβουλων ενεργειών είναι η κατάχρηση του DNS για την εγκαθίδρυση επικοινωνίας ανάμεσα σε μολυσμένες συσκευές (bots) και τους εξυπηρετητές ελέγχου τους (Command & Control ή C&C servers). Συγκεκριμένα, τα bots στηρίζονται στους αλγορίθμους παραγωγής ονομάτων (Domain Generation Algorithms - DGA's) για να παράξουν πλήθος ονομάτων με βάση μία τεχνική seeding, που είναι γνωστή στους C&C servers. Μερικά από τα ονόματα δεσμεύονται, ώστε να αντιστοιχούν στις διευθύνσεις IP των C&C servers, ενώ τα υπόλοιπα ονόματα δε συνδέονται με κάποια πληροφορία DNS. Τα bots ρωτούν τα παραχθέντα ονόματα μέχρι να λάβουν ως απάντηση τη διεύθυνση IP που κατευθύνει στον αντίστοιχο C&C server. Ο συνήθως μεγάλος αριθμός των ερωτηθέντων ονομάτων, καθώς και συχνές αλλαγές στην τεχνική seeding δυσκολεύουν τον εντοπισμό των κακόβουλων ονομάτων και, κατά συνέπεια, το μπλοκάρισμα της λειτουργίας ενός δικτύου μολυσμένων συσκευών (botnet).

Η προστασία του DNS είναι υψίστης σημασίας για τους διαχειριστές δικτύων. Οι μηχανισμοί άμυνας του DNS από τις παραπάνω κυβερνοεπιθέσεις βασίζονται συνήθως σε λίστες καλόβουλων ονομάτων (whitelists) ή αλγορίθμους μηχανικής μάθησης (Machine Learning, ML). Ερωτήματα που περιλαμβάνουν domain names τα οποία περιέχονται στις whitelists προωθούνται στον προορισμό τους, ενώ τα υπόλοιπα απορρίπτονται ως κακόβουλα. Αντίστοιχα, οι αλγόριθμοι μηχανικής μάθησης εκμεταλλεύονται παρελθοντικά δεδομένα για την εκπαίδευση μοντέλων που μπορούν να γενικεύουν σε νέα δεδομένα, ώστε να διακρίνονται αποτελεσματικά τα καλόβουλα από τα κακόβουλα ερωτήματα DNS.

Αν και τα αποτελέσματα των παραπάνω μηχανισμών είναι συνήθως ικανοποιητικά, υπάρχουν σημαντικοί περιορισμοί στη χρήση τους:

- **Περιορισμοί σε υπολογιστικούς πόρους:** Οι παραδοσιακές προσεγγίσεις που χρησιμοποιούν whitelists για να φιλτράρουν επιθετική κίνηση DNS βασίζονται συνήθως σε δομές δεδομένων και αλγορίθμους που απαιτούν την ακριβή μορφή των domain names για να λειτουργήσουν. Whitelists που απεικονίζουν ολόκληρες ζώνες DNS μπορεί να περιλαμβάνουν ακόμα και εκατομμύρια ονόματα. Κατά συνέπεια, η χρήση αναποτε-

λεσματικών τεχνικών μπορεί να δημιουργήσει σημαντικά προβλήματα λόγω περιορισμών στους διαθέσιμους υπολογιστικούς πόρους. Συγκεκριμένα, οι συσκευές που φιλτράρουν την επιθετική κίνηση μπορεί να αδυνατούν να αποθηκεύσουν τις whitelists στη μνήμη τους. Παράλληλα, η απόδοση φιλτραρίσματος των αμυντικών συσκευών μπορεί να μην ανταποκρίνεται σε σύγχρονες επιθέσεις DDoS που πλημμυρίζουν τα θύματά τους με πολύ μεγάλο όγκο κίνησης στη μονάδα του χρόνου.

- **Προβλήματα ιδιωτικότητας:** Η πρόσβαση στα περιεχόμενα των ζωνών που περιλαμβάνουν οι Authoritative DNS Servers είναι συνήθως περιορισμένη για λόγους ασφαλείας. Αυτό εμποδίζει την κατασκευή έγκυρων whitelists που μπορούν να χρησιμοποιηθούν για το ακριβές φιλτράρισμα επιθετικής κίνησης DNS κοντά στις πηγές της. Χαρακτηριστικό παράδειγμα είναι η αντιμετώπιση επιθέσεων DDoS σε Recursors ή upstream υπηρεσίες συννέφου (cloud services), όπου είναι συνήθως πιο αποτελεσματική, καθώς και η κατασκευή whitelists από τα περιεχόμενα διαφόρων Authoritative DNS Servers για το ακριβέστερο φιλτράρισμα κίνησης DNS που παράγεται από DGA's.
- **Περιορισμοί απόδοσης φιλτραρίσματος πακέτων:** Οι μηχανισμοί αντιμετώπισης επιθέσεων DNS βασίζονται συνήθως στη βαθιά επιθεώρηση πακέτου (Deep Packet Inspection, DPI), δηλαδή στην εξέταση των δεδομένων (payload) των μηνυμάτων DNS για την εξαγωγή των domain names. Τα domain names τροφοδοτούνται στη συνέχεια σε whitelists ή αλγορίθμους μηχανικής μάθησης που τα κατηγοριοποιούν ως καλόβουλα ή κακόβουλα. Το DPI και οι κατηγοριοποιήσεις ονομάτων περιλαμβάνουν ενέργειες που πραγματοποιούνται συνήθως στο χώρο χρήστη (user space) των συσκευών φιλτραρίσματος. Ωστόσο, η αντιμετώπιση επιθέσεων στο user space προκαλεί σημαντικές καθυστερήσεις λόγω των συχνών προσβάσεων στη μνήμη, καθώς και άλλων ενεργειών του λειτουργικού συστήματος. Ως αποτέλεσμα, οι μηχανισμοί φιλτραρίσματος μπορεί να αδυνατούν να ανταποκριθούν στον μεγάλο όγκο κίνησης των σύγχρονων επιθέσεων DDoS.
- **Αδυναμία κατανόησης των αποφάσεων των μοντέλων μηχανικής μάθησης:** Οι συνεχείς προσπάθειες για τη βελτίωση της διακριτικής ικανότητας των ταξινομητών μηχανικής μάθησης οδήγησε στην υιοθέτηση όλο και πιο περίπλοκων μοντέλων. Έτσι, απλά και επεξηγήσιμα μοντέλα αντικαταστάθηκαν από πολύπλοκα, μαύρα κουτιά (black-boxes), των οποίων οι αποφάσεις δεν μπορούν να γίνουν άμεσα κατανοητές στους χρήστες των αλγορίθμων και τους προγραμματιστές των μοντέλων. Ως αποτέλεσμα, οι διαχειριστές δικτύων αντιμετωπίζουν με δυσπιστία τη χρήση μοντέλων μηχανικής μάθησης σε συσκευές φιλτραρίσματος δικτυακής κίνησης, καθώς απαιτούν συνήθως διαβεβαιώσεις για τη λειτουργία τους.

Λύσεις για την αντιμετώπιση των παραπάνω περιορισμών μπορούν να αναζητηθούν στο πεδίο των μεγάλων δεδομένων (Big Data) και των δικτύων που καθορίζονται από λογισμικό (Software-Defined Networks, SDN's). Συγκεκριμένα, οι πιθανοτικές δομές δεδομένων (probabilistic data structures), η μηχανική μάθηση και ο προγραμματισμός στο επίπεδο δε-

δομένων (data plane programming) μπορούν να χρησιμοποιηθούν για την αποτελεσματική προστασία από απειλές DNS. Επιπρόσθετα, αλγόριθμοι επεξηγήσιμης τεχνητής νοημοσύνης (eXplainable Artificial Intelligence, XAI) μπορούν να χρησιμοποιηθούν για να ερμηνευτεί η λειτουργία πολύπλοκων μοντέλων μηχανικής μάθησης.

Στη διατριβή αυτή βασιζόμαστε στις παραπάνω τεχνολογίες για την αποτελεσματική προστασία υποδομών DNS από επιθέσεις DDoS, συγκεκριμένα τις επιθέσεις Water Torture, καθώς και το αξιόπιστο φιλτράρισμα μηνυμάτων DNS που παράγονται από DGA's.

Η έρευνά μας στηρίζεται κυρίως:

- Στα φίλτρα Bloom (Bloom Filters, BF's) και τα φίλτρα Cuckoo (Cuckoo Filters, CF's), τα οποία είναι αποδοτικές ως προς τον χώρο αποθήκευσης πιθανοτικές δομές δεδομένων που χρησιμοποιούνται για να εξεταστεί ταχύτατα εάν ένα στοιχείο, π.χ. domain name στην περίπτωσή μας, είναι αποθηκευμένο στο φίλτρο ή όχι. Για να είναι αποδοτικά ως προς τον χρόνο και τη μνήμη, τα BF's και τα CF's δεν αποθηκεύουν τα domain names με την κανονική τους μορφή, αλλά αφού τα περάσουν από συναρτήσεις κατακερματισμού (hash functions). Αφού αλλοιώνεται η αρχική μορφή των δεδομένων, οι παραπάνω πιθανοτικές δομές αποκτούν ιδιότητες ιδιωτικότητας (privacy) και είναι κατάλληλες για την ανταλλαγή ευαίσθητων δεδομένων.
- Στο eXpress Data Path (XDP), μια μέθοδο προγραμματισμού στο επίπεδο δεδομένων που επιτρέπει την επεξεργασία των εισερχόμενων πακέτων σε πολύ υψηλές ταχύτητες. Η επεξεργασία των πακέτων γίνεται στο επίπεδο των οδηγών (drivers) του Linux, μειώνοντας σημαντικά τις συνολικές ενέργειες του λειτουργικού συστήματος που απαιτούνται ανά πακέτο, π.χ. δεσμεύσεις μνήμης. Έναντι άλλων μεθόδων προγραμματισμού στο επίπεδο δεδομένων, βασικά πλεονεκτήματα του XDP είναι ότι δεν απαιτεί εξειδικευμένο υλικό (hardware), όπως το Programming Protocol-independent Packet Processors (P4), ενώ δεν παρακάμπτει τον πυρήνα του Linux (Linux kernel), όπως το Data Plane Development Kit (DPDK), επιτρέποντας τη χρήση λειτουργιών που είναι διαθέσιμες από τον Linux kernel, π.χ. τις βιβλιοθήκες της στοίβας TCP/IP.
- Στην τεχνική SHapley Additive exPlanation (SHAP), μία μέθοδο XAI που επιτρέπει την εξαγωγή καθολικών (global) και τοπικών (local) ερμηνειών (interpretations) για τη λειτουργία πολύπλοκων μοντέλων μηχανικής μάθησης. Οι καθολικές ερμηνείες αφορούν ομάδες δειγματικών σημείων του συνόλου δεδομένων (dataset), ενώ οι τοπικές αφορούν μεμονωμένα δειγματικά σημεία. Οι ερμηνείες που προσφέρει η SHAP μπορούν να ληφθούν για οποιοδήποτε μοντέλο (model-agnostic), αφού ολοκληρωθεί η εκπαίδευσή του (post-hoc).

Οι βασικές συνεισφορές της διατριβής είναι οι παρακάτω:

- **Ανίχνευση και αντιμετώπιση επιθέσεων Water Torture με αποτελεσματικές δομές δεδομένων και αποδοτικούς αλγορίθμους:** Αξιοποιούμε τεχνικές από το πεδίο των Big Data για την αποτελεσματική προστασία απέναντι σε επιθέσεις που πλήττουν

υποδομές DNS, συγκεκριμένα τις επιθέσεις Water Torture. Τα προτεινόμενα συστήματα βασίζονται σε πιθανοτικές δομές δεδομένων (Bloom Filters - BF's και Count-Min Sketches - CMS's) για να πραγματοποιήσουν ελέγχους ονομάτων (name lookups) και εκτίμηση συχνοτήτων (frequency estimation) αποδοτικά ως προς τον χρόνο επεξεργασίας και τη μνήμη που καταναλώνεται. Παράλληλα, στηριζόμαστε σε αιτιοκρατικούς (deterministic) αλγορίθμους επεξεργασίας φυσικής γλώσσας (Natural Language Processing, NLP), όπως είναι ο SymSpell, για τον ταχύτατο εντοπισμό τυπογραφικών λαθών, ο οποίος επιτρέπει τη λεπτομερή κατηγοριοποίηση διευθύνσεων IP σε καλόβουλες και κακόβουλες. Σε αντίθεση με παρόμοιες προσεγγίσεις, οι προτεινόμενοι μηχανισμοί μπορούν να απεικονίζουν μεγάλες ζώνες DNS χωρίς μείωση της συνολικής απόδοσης με κόστος την εισαγωγή ενός ανεκτού και παραμετροποιήσιμου ποσοστού λαθών.

- **Επιτάχυνση ενεργειών DPI με προγραμματισμό στο επίπεδο δεδομένων για την αποτελεσματική αντιμετώπιση επιθέσεων DNS:** Βασιζόμαστε στο XDP για να ξεχωρίσουμε καλόβουλα και κακόβουλα μηνύματα DNS στο επίπεδο δεδομένων (data plane) των συσκευών φιλτραρίσματος δικτυακής κίνησης. Το XDP χρησιμοποιείται για τη βελτίωση ενεργειών DPI που παραδοσιακά πραγματοποιούνται στο χώρο χρήση των συστημάτων ασφαλείας, δηλαδή την εξαγωγή των ερωτηθέντων ονομάτων, ελέγχους σε whitelists, καθώς και την αποτίμηση της εξόδου αλγορίθμων μηχανικής μάθησης. Για την υλοποίηση των παραπάνω ενεργειών DPI στο data plane, λαμβάνονται υπόψη οι περιορισμοί του XDP, δηλαδή η απουσία δεκαδικών αριθμών, το περιορισμένο μέγεθος προγράμματος, αλλά και η έλλειψη ατέρμονων βρόχων. Οι data plane υλοποιήσεις μας αυξάνουν σημαντικά την απόδοση φιλτραρίσματος σε σχέση με τις αντίστοιχες υλοποιήσεις σε χώρο χρήστη, χωρίς να απαιτείται εξειδικευμένο hardware, όπως σε παρόμοιες μεθόδους προγραμματισμού στο επίπεδο δεδομένων, π.χ. το P4.
- **Ανταλλαγές ζωνών DNS που σέβονται την ιδιωτικότητα βασισμένες σε πιθανοτικές δομές δεδομένων:** Βασιζόμαστε στις ιδιότητες ιδιωτικότητας των πιθανοτικών δομών δεδομένων και συγκεκριμένα στα Cuckoo Filters (CF's) για την ανταλλαγή ζωνών DNS ανάμεσα σε Authoritative DNS Servers και συσκευές φιλταρίσματος εξωτερικών συνεργατών (third-parties), π.χ. Recursors ή upstream υποδομές συννέφου (cloud infrastructures). Έτσι, η αντιμετώπιση επιθέσεων DDoS που πλήττουν το DNS (π.χ. Water Torture) μπορεί να πραγματοποιηθεί πιο κοντά στις πηγές της επίθεσης, όπου το φιλτράρισμα δικτυακής κίνησης είναι συνήθως πιο αποδοτικό. Παράλληλα, οι διαχειριστές δικτύων μπορούν να συλλέξουν whitelists από διάφορους Authoritative DNS Servers ωστε να φιλτράρουν με αυξημένη ακρίβεια κίνηση DNS που παράγεται από DGA's στις άκρες των δικτύων τους. Σε αντίθεση με παρόμοιες προσεγγίσεις που βασίζονται σε BF's, χρησιμοποιούμε CF's, τα οποία επιτρέπουν τις διαγραφές στοιχείων και, κατά συνέπεια, διευκολύνουν την ανανέωση των whitelists.
- **Καθολικές και τοπικές ερμηνείες ταξινομητών DGA που βασίζονται σε δέντρα αποφάσεων και βαθιά νευρωνικά δίκτυα με χρήση model-agnostic, post-hoc μεθό-**

δων XAI: Βασιζόμαστε στην τεχνική SHAP για να ερμηνεύσουμε τις αποφάσεις ταξινομητών ονομάτων που παράγονται από DGA's με model-agnostic και post-hoc τρόπο, δηλαδή ανεξάρτητα από το επιλεγμένο μοντέλο και αφού ολοκληρωθεί η διαδικασία μάθησης. Αξιοποιούμε πλήθος εργαλείων οπτικοποίησης που παρέχει η μέθοδος SHAP (διαγράμματα summary, dependence, force) για την εξαγωγή καθολικών και τοπικών ερμηνειών των κριτηρίων με τα οποία οι ταξινομητές ονομάτων διακρίνουν τα καλόβουλα από τα κακόβουλα ονόματα. Σε αντίθεση με παρόμοιες προσεγγίσεις, εκτιμούμε τη συνεισφορά των επιλεγμένων χαρακτηριστικών (features) και τις αλληλεπιδράσεις τους εστιάζοντας στην τεχνική που έχει χρησιμοποιηθεί για την παραγωγή των ονομάτων DGA (arithmetic, wordlist, hash και permutation based). Η εκπαίδευση των αλγορίθμων μηχανικής μάθησης και η εξαγωγή ερμηνειών βασίζονται σε features που εξάγονται απευθείας από τα domain names. Τα features αυτά αποδίδουν τις στατιστικές και γλωσσικές ιδιότητες των ονομάτων, ενώ αποφεύγονται λήψεις δεδομένων από εξωτερικές βάσεις δεδομένων. Τέτοιες ενέργειες είναι συνήθως χρονοβόρες και μπορεί να παραβιάσουν την ιδιωτικότητα διαφόρων δικτυακών χρηστών.

- **Εκτενής πειραματική αξιολόγηση των προτεινόμενων μηχανισμών:** Τα προτεινόμενα συστήματα ασφαλείας αξιολογούνται με πειράματα στην εικονική υποδομή του εργαστηρίου Διαχείρισης και Βέλτιστου Σχεδιασμού Δικτύων Τηλεματικής στο Εθνικό Μετσόβιο Πολυτεχνείο (ΕΜΠ). Οι αξιολογήσεις μας βασίζονται σε έγκυρα σύνολα δεδομένων που χρησιμοποιούνται κατά κόρον από τους ερευνητές DNS (π.χ. DGArchive, λίστα Tranco) και συνθετική κίνηση, η οποία βασίστηκε σε στατιστικά στοιχεία από δημόσια προσβάσιμα δεδομένα (επιθέσεις Booters) και την εμπειρία μας από τις δικτυακές υπηρεσίες του ΕΜΠ.

10.2 Κεφάλαιο 2

Το κεφάλαιο αυτό περιλαμβάνει θεωρητικό υπόβαθρο για έννοιες που είναι σημαντικές για την κατανόηση των υπολοίπων κεφαλαίων της διατριβής. Το κεφάλαιο είναι χωρισμένο σε 7 ενότητες:

- Στην πρώτη ενότητα παρέχονται βασικές πληροφορίες για το σύστημα ονοματοδοσίας (Domain Name System, DNS). Αρχικά, το DNS αναλύεται ως κατανεμημένη και ιεραρχική βάση δεδομένων και, στη συνέχεια, περιγράφονται η διαδικασία επίλυσης ονομάτων, η προσωρινή αποθήκευση ονομάτων στους Recursors, οι τύποι εγγραφών DNS και οι μεταφορές ζωνών. Τέλος, η ενότητα περιλαμβάνει πληροφορίες για τα μηνύματα DNS και τις επικεφαλίδες τους.
- Η δεύτερη ενότητα περιλαμβάνει θεωρητικό υπόβαθρο για τις επιθέσεις DDoS. Αναλύονται τεχνικές, όπως η παραποίηση διευθύνσεων IP (IP spoofing), ενώ παρέχονται πληροφορίες για τα δίκτυα μολυσμένων συσκευών (botnets). Στη συνέχεια, περιγράφονται οι βασικές κατηγορίες επιθέσεων DDoS, καθώς και κοινές τεχνικές εντοπισμού

και αντιμετώπισής τους.

- Στην τρίτη ενότητα παρέχονται πληροφορίες για δικτυακές απειλές που σχετίζονται με το DNS. Αρχικά, περιγράφονται οι επιθέσεις DDoS που στοχεύουν το DNS ή βασίζονται σε αυτό (DNS Flood, Water Torture και Amplification). Επιπρόσθετα, περιλαμβάνονται πληροφορίες για τους Domain Generation Algorithms (DGA's).
- Η τέταρτη ενότητα αφορά SDN's και μεθόδους προγραμματισμού στο επίπεδο δεδομένων (data plane programming). Αρχικά, παρέχονται βασικές πληροφορίες για τα SDN's και το πρωτόκολλο OpenFlow (OF). Στη συνέχεια, περιγράφονται τρεις τεχνικές data plane programming: (i) P4, (ii) XDP και (iii) DPDK.
- Η πέμπτη ενότητα αναλύει πρωτόκολλα και εργαλεία για την παρακολούθηση δικτυακών υποδομών, όπως είναι το NetFlow και το sFlow.
- Η έκτη ενότητα περιγράφει αποδοτικές δομές δεδομένων και αλγορίθμους που χρησιμοποιούνται στα υπόλοιπα κεφάλαια της διατριβής. Αναλύονται πιθανοτικές δομές δεδομένων, συγκεκριμένα τα Bloom Filters (BF's), τα Cuckoo Filters (CF's) και τα Count-Min Sketches (CMS's). Έπειτα, παρέχονται πληροφορίες για αλγορίθμους διόρθωσης τυπογραφικών λαθών με έμφαση στον αλγόριθμο SymSpell.
- Στην τελευταία ενότητα συνοψίζονται βασικές έννοιες της μηχανικής μάθησης, περιγράφονται βασικοί αλγόριθμοι μηχανικής μάθησης που χρησιμοποιούνται στα υπόλοιπα κεφάλαια της διατριβής και παρέχονται πληροφορίες για τεχνικές επεξηγήσιμης τεχνητής νοημοσύνης (eXplainable Artificial Intelligence, XAI).

10.3 Κεφάλαιο 3

Στο κεφάλαιο αυτό παρουσιάζεται ο μηχανισμός που αναπτύχθηκε για την αποτελεσματική ανίχνευση και αντιμετώπιση επιθέσεων Water Torture. Βασικό ρόλο έχουν δομές δεδομένων και αλγόριθμοι που εξασφαλίζουν χαμηλό χρόνο επεξεργασίας πακέτων και μειώνουν τη συνολική απαιτούμενη μνήμη. Συγκεκριμένα, ο προτεινόμενος μηχανισμός στηρίζεται σε πιθανοτικές δομές δεδομένων (Bloom Filters – BF's, Count-Min Sketches – CMS's) και σε αιτιοκρατικούς αλγορίθμους για τη διόρθωση τυπογραφικών λαθών (SymSpell). Η λειτουργία των παραπάνω τεχνικών γίνεται στο χώρο χρήστη (user space).

Η κύρια συνεισφορά εντοπίζεται στην αξιοποίηση των BF's για την ακριβής απεικόνιση μεγάλων ζωνών DNS. Στα BF's αποθηκεύουμε, οικονομικά ως προς την απαιτούμενη μνήμη, τα ονόματα των ζωνών του Authoritative DNS Server που θέλουμε να προστατέψουμε από τις επιθέσεις Water Torture. Έπειτα, χρησιμοποιούμε τα BF's για τον αποδοτικό διαχωρισμό καλόβουλης και κακόβουλης κίνησης DNS, καθώς επιτρέπουν ταχύτατους ελέγχους για να εκτιμηθεί εάν ένα όνομα είναι αποθηκευμένο στο φίλτρο ή όχι.

Ένα βασικό χαρακτηριστικό των BF's, που τα καθιστά κατάλληλα για την άμυνα απέναντι σε επιθέσεις Water Torture, είναι η απουσία ψευδώς αρνητικών (False Negatives, FN's) αποτελεσμάτων. Η απουσία FN's εξασφαλίζει ότι κανένα καλόβουλο όνομα δεν ταξινομείται λανθασμένα. Επομένως, το σύνολο των χρήσιμων μηνυμάτων DNS οδηγείται στον

Authoritative DNS Server για να εξυπηρετηθεί. Ωστόσο, τα BF's μπορεί να επιστρέψουν ψευδώς θετικά (False Positives, FP's) αποτελέσματα, δηλαδή ένα μικρό μέρος από την επιθετική κίνηση της Water Torture θα προωθηθεί στο θύμα. Η πιθανότητα εμφάνισης ενός FP καθορίζεται από τις παραμέτρους των BF's.

Ένα ακόμα μεγάλο πλεονέκτημα των BF's αποτελούν και οι ιδιότητες ιδιωτικότητας (privacy) που παρέχουν. Η αρχική μορφή των ονομάτων, που περιλαμβάνονται στις ζώνες του Authoritative DNS Server, αλλοιώνεται με την εφαρμογή συναρτήσεων κατακερματισμού που απαιτούνται για την αποθήκευση δεδομένων στο BF. Έτσι, ευνοείται η ανάπτυξη συνεργατικών σχημάτων αντιμετώπισης επιθέσεων Water Torture, καθώς τα BF's διευκολύνουν τον διαμοιρασμό ζωνών σε εξωτερικούς συνεργάτες (π.χ. Recursors, cloud services) χωρίς την αποκάλυψη ευαίσθητων πληροφοριών. Οι εξωτερικοί συνεργάτες μπορούν να φιλτράρουν την κακόβουλη δικτυακή κίνηση πιο κοντά στις πηγές της, όπου συνήθως ο έλεγχος της επίθεσης είναι πιο αποτελεσματικός.

Ο προτεινόμενος μηχανισμός παρουσιάζεται στο σχήμα 3.1. Αποτελείται από τα ακόλουθα τρία δομικά στοιχεία (components): (α) Anomaly Detection Component (ADC), (β) Mitigation Trigger Component (MTC) και (γ) DNS FireWall (DFW). Το ADC συλλέγει δεδομένα παρακολούθησης από το δίκτυο του θύματος (Authoritative DNS Server), εντοπίζει επιθέσεις Water Torture και κατηγοριοποιεί τις διευθύνσεις IP ως καλόβουλες ή ύποπτες. Για τις ύποπτες διευθύνσεις IP, το ADC διατυπώνει αιτήματα αντιμετώπισης προς το MTC. Στη συνέχεια, το MTC προβαίνει στις αναγκαίες ενέργειες, ώστε η κίνηση DNS που προέρχεται από τις ύποπτες διευθύνσεις IP να κατευθυνθεί προς το DFW, όπου θα φιλτραριστεί.

Τα DFW's περιλαμβάνουν BF's που ελέγχουν τα εισερχόμενα μηνύματα DNS. Μηνύματα που αναγνωρίζονται ως κακόβουλα απορρίπτονται, ενώ τα υπόλοιπα (καλόβουλα μηνύματα και μερικά κακόβουλα που δεν εντοπίστηκαν λόγω των FP's του BF) προωθούνται στον Authoritative DNS Server για εξυπηρέτηση. Το DFW λειτουργεί είτε στο δίκτυο του θύματος ή σε κάποια υπηρεσία συννέφου. Τέλος, το MTC ενημερώνει τους Recursors που προωθούν κακόβουλη κίνηση μέσω ενός συνεργατικού σχήματος. Οι Recursors που είναι πρόθυμοι να φιλτράρουν την επιθετική κίνηση κοντά στις πηγές της θα λάβουν τα περιεχόμενα των ζωνών αποθηκευμένα σε BF's, χωρίς να αποκαλυφθούν ευαίσθητα δεδομένα.

Η αξιολόγηση του προτεινόμενου μηχανισμού εστιάζεται σε τρία σκέλη:

- Αρχικά, ερευνούμε εάν τα BF's απεικονίζουν ικανοποιητικά μεγάλες ζώνες. Εκτιμούμε την απαιτούμενη μνήμη για την επίτευξη μίας δεδομένης πιθανότητας ψευδώς θετικών αποτελεσμάτων, όταν απεικονίζεται συγκεκριμένος αριθμός ονομάτων. Η ανάλυσή μας βασίζεται στη ζώνη "ru" που περιλαμβάνει περίπου 5 εκατομμύρια ονόματα και αποτελεί μία από τις 10 μεγαλύτερες ζώνες. Οι υπολογισμοί μας δείχνουν ότι ένα BF μεγέθους 16.95 MB μπορεί να αναπαραστήσει ικανοποιητικά τη ζώνη με ποσοστό FP 0.01%, δηλαδή μία λανθασμένη απόφαση στα 10,000 μηνύματα.
- Στη συνέχεια, αξιολογούμε την ικανότητα του ADC να ανιχνεύει έγκαιρα επιθέσεις Water Torture και ύποπτες διευθύνσεις IP. Η αξιολόγηση στηρίχθηκε σε συνθετικά δεδομένα καλόβουλης και κακόβουλης δικτυακής κίνησης που βασίστηκαν σε δημο-

σίως προσβάσιμα datasets. Τα καλόβουλα και κακόβουλα πακέτα προωθήθηκαν ταυτόχρονα στον Authoritative DNS Server με ρυθμούς 3 και 100 Kpps αντίστοιχα. Το ADC εντόπισε τις διευθύνσεις IP που προωθούσαν το μεγαλύτερο μέρος της κακόβουλης δικτυακής κίνησης σε λιγότερο από ένα λεπτό. Παρατηρήσαμε ότι καμία καλόβουλη διεύθυνση IP δεν κατηγοριοποιήθηκε λανθασμένα λόγω της απουσίας FN's στα BF's. Παράλληλα, η μνήμη που χρησιμοποιήθηκε για τις πιθανοτικές δομές δεδομένων (BF, CMS) και τον αλγόριθμο SymSpell ήταν πολύ μικρή.

- Στο τελευταίο σκέλος αξιολογήσαμε την αποτελεσματικότητα των BF's για το φιλτράρισμα επιθέσεων Water Torture. Συγκρίναμε τα BF's με τα Red-Black Trees (RBT's), τα οποία χρησιμοποιούνται από το εδραιωμένο λογισμικό DNS ανοιχτού κώδικα BIND για τη διατήρηση ζωνών στη μνήμη των εξυπηρετητών DNS. Προωθώντας μηνύματα DNS ταυτόχρονα στα BF's και τα RBT's παρατηρήσαμε ότι τα BF's φιλτράρουν αποδοτικότερα την επιθετική κίνηση όσο μεγαλώνει το μέγεθος της αποθηκευμένης ζώνης.

Ο προτεινόμενος μηχανισμός προστάτεψε αποτελεσματικά υποδομές DNS από επιθέσεις Water Torture. Ωστόσο, η υλοποίηση του μηχανισμού αντιμετώπισης (που βασίζεται σε BF's) στο χώρο χρήστη περιορίζει σημαντικά τις δυνατότητες φιλτραρίσματος δικτυακής κίνησης. Στο επόμενο κεφάλαιο, χρησιμοποιούμε προγραμματισμό σε επίπεδο δεδομένων για τη βελτίωση της απόδοσης του μηχανισμού αντιμετώπισης επιθέσεων Water Torture.

10.4 Κεφάλαιο 4

Στο κεφάλαιο αυτό αξιοποιούμε το eXpress Data Path (XDP) για την αποτελεσματική αντιμετώπιση επιθέσεων Water Torture στο επίπεδο δεδομένων (data plane) των Authoritative DNS Servers. Αντίστοιχα με το προηγούμενο κεφάλαιο, τα BF's χρησιμοποιούνται για την απεικόνιση ζωνών DNS και το φιλτράρισμα δικτυακής κίνησης με τρόπο αποδοτικό ως προς τον χρόνο (έλεγχος ύπαρξης ονομάτων) και τη μνήμη που καταναλώνεται.

Βασικές συνεισφορές του προτεινόμενου μηχανισμού είναι η (α) αντιμετώπιση επιθέσεων DDoS αποκλειστικά εντός του πυρήνα του Linux, (β) διερεύνηση των δυνατοτήτων DPI του XDP για το φιλτράρισμα επιθέσεων DNS επιπέδου εφαρμογής (application layer) και (γ) επίτευξη υψηλής ταχύτητας αντιμετώπισης επιθέσεων DDoS χωρίς απαιτήσεις για εξειδικευμένο hardware σε αντίθεση με παραπλήσιες τεχνικές προγραμματισμού στο επίπεδο δεδομένων, όπως το P4.

Ο προτεινόμενος μηχανισμός παρουσιάζεται στο σχήμα 4.1. Το XDP απομονώνει τα εισερχόμενα πακέτα στο επίπεδο των οδηγών (drivers) του Authoritative DNS Server. Κάθε πακέτο προκαλεί την εκτέλεση ενός προγράμματος extended Berkeley Packet Filter (eBPF program). Το πρόγραμμα eBPF ελέγχει, αρχικά, εάν το πακέτο αφορά ερώτημα DNS ή όχι. Απαντήσεις DNS ή πακέτα που δε σχετίζονται με το DNS προωθούνται στο χώρο χρήστη χωρίς επιπρόσθετη επεξεργασία. Στη συνέχεια, εξάγεται το όνομα που περιλαμβάνεται στο ερώτημα DNS και διέρχεται από πλήθος συναρτήσεων κατακερματισμού για να ελεγχθεί εάν το όνομα είναι αποθηκευμένο στο BF ή όχι. Εάν το όνομα δεν είναι αποθηκευμένο στο

BF, το ερώτημα DNS απορρίπτεται εντός του πυρήνα του Linux ως κακόβουλο. Αντίθετα, τα ερωτήματα που χαρακτηρίζονται ως καλόβουλα προωθούνται στο χώρο χρήστη, όπου θα τα διαχειριστεί το λογισμικό DNS. Η απόρριψη των κακόβουλων πακέτων σε χαμηλό επίπεδο μειώνει δραστικά τον αριθμό των ενεργειών του λειτουργικού συστήματος που απαιτούνται για τα πακέτα αυτά και, κατά συνέπεια, αυξάνει σημαντικά την αποτελεσματικότητα φιλτραρίσματος δικτυακής κίνησης.

Η αξιολόγηση του μηχανισμού βασίζεται σε δύο μέρη. Το πρώτο μέρος περιλαμβάνει την εκτίμηση κρίσιμων σχεδιαστικών παραμέτρων του μηχανισμού (π.χ. μέγεθος BF), ενώ το δεύτερο μέρος μελετά την αποτελεσματικότητα του προγράμματος eBPF στην αντιμετώπιση επιθέσεων Water Torture. Η αξιολόγηση εστιάζει στην επαλήθευση του trade-off ανάμεσα στο μέγεθος του BF και τον αριθμό των επιλεγμένων συναρτήσεων κατακερματισμού, που συνδυαστικά καθορίζουν την απόδοση του προγράμματος eBPF. Αύξηση του αριθμού των συναρτήσεων κατακερματισμού μέχρι μία δεδομένη τιμή συνεπάγεται μείωση στο μέγεθος του BF, αλλά υποβαθμίζει την αποτελεσματικότητα του προγράμματος eBPF.

Αρχικά διερευνάται εάν ο προτεινόμενος μηχανισμός περιορίζεται σημαντικά από τους κανόνες που διέπουν τη λειτουργία των προγραμμάτων eBPF. Συγκεκριμένα, ελέγχουμε κατά πόσο το eBPF περιορίζει το μέγιστο μέγεθος ονομάτων που μπορούν να υποστηριχθούν στην εφαρμογή μας. Η ανάλυση καταλήγει ότι οι περιορισμοί είναι μικροί και πως η πλειοψηφία των domain names μπορεί να υποστηριχθεί από το μηχανισμό μας.

Στη συνέχεια, διερευνάται η απόδοση του προγράμματος eBPF. Προωθώντας ταυτόχρονα καλόβουλη και κακόβουλη δικτυακή κίνηση σε έναν Authoritative DNS Server παρατηρούμε ότι η χρήση προγραμματισμού σε επίπεδο δεδομένων βελτιώνει την αποτελεσματικότητα του μηχανισμού αντιμετώπισης επιθέσεων Water Torture (που βασίζεται σε BF's) περίπου 7 φορές σε σχέση με την αντίστοιχη υλοποίηση του μηχανισμού σε χώρο χρήστη.

Αν και η απόδοση του μηχανισμού βελτιώθηκε σημαντικά με την υλοποίηση στο επίπεδο δεδομένων των Authoritative DNS Servers, η αντιμετώπιση επιθέσεων DDoS, όπως η Water Torture, είναι συνήθως αποτελεσματικότερη κοντά στις πηγές τους. Για το σκοπό αυτό, στο επόμενο κεφάλαιο υλοποιούμε έναν μηχανισμό για την ανταλλαγή ζωνών DNS με τρόπο που σέβεται την ιδιωτικότητα.

10.5 Κεφάλαιο 5

Στο κεφάλαιο αυτό παρουσιάζεται ένας μηχανισμός για την αποτελεσματική διανομή ζωνών DNS από Authoritative DNS Servers. Ο προτεινόμενος μηχανισμός βασίζεται σε μεθόδους που αποκρύπτουν τα περιεχόμενα των ζωνών, ευνοώντας την αξιοποίηση των ληφθέντων πληροφοριών από εξωτερικές συσκευές φιλτραρίσματος δικτυακής κίνησης. Τέτοιες συσκευές μπορεί να περιλαμβάνονται σε Recursors ή upstream υποδομές συννέφου. Πιθανές εφαρμογές του προτεινόμενου μηχανισμού αποτελούν η αντιμετώπιση επιθέσεων Water Torture πιο κοντά στις πηγές της επίθεσης (π.χ. σε Recursors) και το ακριβέστερο φιλτράρισμα κίνησης DNS για την απόρριψη ερωτημάτων που παράγονται από DGA's.

Καθορίζουμε διάφορες σχεδιαστικές απαιτήσεις για τον προτεινόμενο μηχανισμό. Συγκεκριμένα, ο μηχανισμός πρέπει να απεικονίζει τα ονόματα που περιλαμβάνονται στις ζώνες DNS αποδοτικά και με τρόπο που εξασφαλίζει την ιδιωτικότητά τους. Επιπρόσθετα, η λειτουργία του μηχανισμού απαιτείται να είναι συμβατή με υπάρχοντα εργαλεία DNS (ερωτήματα AXFR και IXFR), καθώς και να υποστηρίζονται ανανεώσεις των διαθέσιμων πληροφοριών χωρίς την εκ νέου λήψη ολόκληρων των ζωνών.

Για την ικανοποίηση των παραπάνω προϋποθέσεων, τα περιεχόμενα των ζωνών απεικονίζονται σε Cuckoo Filters (CF's). Παρόμοια με τα BF's, τα CF's διαθέτουν ιδιότητες ιδιωτικότητας, καθώς τα ονόματα διέρχονται από συναρτήσεις κατακερματισμού πριν την αποθήκευσή τους στη δομή, χάνοντας έτσι την αρχική τους μορφή. Ωστόσο, τα CF's είναι πιο αποδοτικά από τα BF's ως προς τον απαιτούμενο χρόνο αναζήτησης στοιχείων και τη μνήμη που καταναλώνεται, ενώ σε αντίθεση με τα BF's, τα CF's υποστηρίζουν διαγραφές ονομάτων που έχουν εισαχθεί προηγουμένως στο φίλτρο.

Ο προτεινόμενος μηχανισμός παρουσιάζεται στο σχήμα 5.1. Κύριες συνιστώσες του μηχανισμού είναι: (α) οι plaintext ζώνες DNS (Plaintext DNS Zones, PltZn's), που περιλαμβάνουν τα ονόματα των ζωνών με την κανονική τους μορφή, (β) οι hashed ζώνες DNS (Hashed DNS Zones, HsZn's), που έχουν αποθηκευμένα όλα τα ονόματα των ζωνών σε μορφή που σέβεται την ιδιωτικότητα, δηλαδή ως στοιχεία ενός CF, (γ) τις incremental ζώνες DNS (Incremental DNS Zones, IncZn's), που διαθέτουν πρόσφατες αλλαγές των PltZn's σε μορφή που σέβεται την ιδιωτικότητα και (δ) τον Privacy-Aware Zone Manager (PAZM), που κατασκευάζει και διαχειρίζεται τις HsZn's/IncZn's.

Βασικό μέρος της αξιολόγησης του προτεινόμενου μηχανισμού είναι η επικύρωση των δυνατοτήτων του να αποκρύπτει τα περιεχόμενα των ληφθέντων ζωνών DNS. Αν και τα CF's αποθηκεύουν τα διαθέσιμα ονόματα αφού τα περάσουν από συναρτήσεις κατακερματισμού, οι επιτιθέμενοι μπορεί να επιχειρήσουν να ανακαλύψουν περισσότερα για τα περιεχόμενα των ζωνών εκτελώντας επιθέσεις ωμής βίας (brute-force attacks). Οι επιθέσεις brute-force δοκιμάζουν όλους τους δυνατούς συνδυασμούς επιτρεπτών χαρακτήρων και καταγράφουν τις θετικές απαντήσεις του CF.

Η αξιολόγησή μας περιλαμβάνει ένα CF που απεικονίζει τη ζώνη του Εθνικού Μετσοβίου Πολυτεχνείου (8,303 ονόματα). Εκτελώντας επίθεση brute-force στο CF, καταγράφουμε τις αληθώς θετικές (True Positives, TP's) και ψευδώς θετικές (False Positives, FP's) απαντήσεις μεγαλώνοντας σταδιακά το πρόθεμα (prefix) των domain names που ερωτούνται. Για μικρά prefixes (3-4 χαρακτήρες), παρατηρούμε ότι ο αριθμός των TP's και FP's είναι παραπλήσιος. Επομένως, οι επιτιθέμενοι μπορούν να ανακαλύψουν πληροφορίες για τη ζώνη παρότι τα ονόματα δε βρίσκονται στην κανονική τους μορφή. Ωστόσο, για μεγαλύτερα μήκη των prefixes, ο αριθμός των FP's ξεπερνά σημαντικά τον αριθμό των TP's. Συγκεκριμένα, ο λόγος FP's/TP's αυξάνεται δραστικά, καθώς αυξάνεται το μήκος των εξεταζόμενων prefixes. Ως αποτέλεσμα, τα περιεχόμενα των ζωνών αποκρύπτονται με μεγάλη βεβαιότητα.

Οι μηχανισμοί που παρουσιάστηκαν μέχρι αυτό το σημείο βασίζονταν σε λίστες καλόβουλων ονομάτων. Όμως, οι διαχειριστές δικτύων μπορεί να είναι απρόθυμοι να μοι-

ραστούν τα δεδομένα των εξυπηρετητών τους. Σε αυτήν την περίπτωση μπορεί να γίνει χρήση μεθόδων ανίχνευσης ανωμαλιών, π.χ. αλγόριθμοι μηχανικής μάθησης. Η αποτίμηση των εξόδων των αλγορίθμων αυτών υλοποιείται κυρίως στο επίπεδο χρήστη (user space) με αποτέλεσμα η απόδοση φιλτραρίσματος μηνυμάτων DNS να μην ανταποκρίνεται στον όγκο των σύγχρονων επιθέσεων DDoS. Στο επόμενο κεφάλαιο χρησιμοποιούμε data plane programming για την ταχύτατη αντιμετώπιση επιθέσεων DNS μέσω της αποτίμησης των εξόδων αλγορίθμων ML στο επίπεδο δεδομένων των Recursors.

10.6 Κεφάλαιο 6

Στο κεφάλαιο αυτό χρησιμοποιούμε προγραμματισμό σε επίπεδο δεδομένων για τη βελτίωση της απόδοσης μηχανισμών αντιμετώπισης δικτυακών επιθέσεων που βασίζονται σε αλγορίθμους μηχανικής μάθησης. Συγκεκριμένα, αξιοποιούμε το XDP για να υλοποιήσουμε στο data plane χρονικά κρίσιμες ενέργειες, όπως είναι η εξαγωγή των χαρακτηριστικών (features) και η αποτίμηση των εξόδων δυαδικών ταξινομητών. Αντίθετα, πολύπλοκες και χρονοβόρες ενέργειες, όπως είναι η εκπαίδευση μοντέλων μηχανικής μάθησης, πραγματοποιούνται στον χώρο χρήστη.

Ο προτεινόμενος μηχανισμός μπορεί να εφαρμοστεί σε διάφορες δικτυακές επιθέσεις. Ως use case επιλέξαμε την αντιμετώπιση επιθέσεων Water Torture σε αναδρομικούς εξυπηρετητές DNS (Recursors). Ως αλγόριθμο μηχανικής μάθησης επιλέξαμε τον Naive Bayes για δυαδική ταξινόμηση δικτυακής κίνησης, δηλαδή για τον διαχωρισμό καλόβουλων και κακόβουλων ερωτημάτων DNS. Ο Naive Bayes επιλέχθηκε επειδή είναι απλός, ακριβής και αποδοτικός ως προς τον χρόνο απόφασης αλγόριθμος μηχανικής μάθησης.

Η εξαγωγή των features γίνεται από το domain name και η επιλογή τους βασίζεται στην καταλληλότητά τους να ξεχωρίζουν έγκυρα από άκυρα ονόματα. Η προσέγγισή μας περιλαμβάνει 7 features. Αυτά είναι το μήκος του προθέματος (prefix) του ονόματος, ο αριθμός και το μήκος της μέγιστης ακολουθίας φωνηέντων στο prefix, ο αριθμός και το μήκος της μέγιστης ακολουθίας συμφώνων στο prefix, καθώς και ο αριθμός και το μήκος της μέγιστης ακολουθίας δεκαδικών ψηφίων στο prefix.

Ο προτεινόμενος μηχανισμός παρουσιάζεται στο σχήμα 6.1. Τα εισερχόμενα πακέτα απομονώνονται από το XDP στο επίπεδο των drivers του Recursor. Κάθε πακέτο προκαλεί την εκτέλεση ενός προγράμματος eBPF. Αρχικά, το πρόγραμμα ελέγχει εάν το πακέτο περιλαμβάνει ένα ερώτημα DNS. Πακέτα που δεν αφορούν ερωτήματα DNS προωθούνται στο χώρο χρήστη, όπου θα τα επεξεργαστεί η αντίστοιχη εφαρμογή. Αντίθετα, τα ερωτήματα DNS θα υποστούν επιπρόσθετη επεξεργασία. Συγκεκριμένα, το πρόγραμμα eBPF εξάγει το όνομα από το πεδίο ερωτήσεων και υπολογίζει τα features που θα δοθούν ως είσοδοι στον ταξινομητή Naive Bayes. Έπειτα, ο Naive Bayes υπολογίζει την πιθανότητα να είναι το πακέτο καλόβουλο και την πιθανότητα να είναι κακόβουλο. Ανάλογα με το ποια πιθανότητα είναι μεγαλύτερη, το ερώτημα DNS ταξινομείται στην αντίστοιχη κατηγορία. Ερωτήματα DNS που χαρακτηρίζονται ως καλόβουλα προωθούνται στον χώρο χρήστη για να τα επεξεργαστεί το λογισμικό DNS. Αντίθετα, τα μηνύματα που κατηγοριοποιούνται ως κακόβουλα

απορρίπτονται εντός του Linux kernel, οπότε το λογισμικό DNS δεν επιβαρύνεται.

Σημαντική πρόκληση στην υλοποίηση του μηχανισμού αποτελούν οι περιορισμοί που θέτει το XDP, δηλαδή το περιορισμένο μέγεθος προγράμματος, η απουσία ατέρμονων βρόχων και η έλλειψη υποστήριξης δεκαδικών αριθμών. Από τους παραπάνω περιορισμούς, ο τελευταίος είναι ο σημαντικότερος, καθώς ο αλγόριθμος Naive Bayes βασίζεται σε δεκαδικούς αριθμούς για τη λειτουργία του. Για να ξεπεράσουμε τον περιορισμό, πολλαπλασιάζουμε τις πιθανότητες του Naive Bayes με δυνάμεις του 10 και τις απεικονίζουμε ως δεκαδικούς αριθμούς. Το ακέραιο μέρος των αριθμών διατηρείται, ενώ το δεκαδικό αποκόπτεται. Ως αποτέλεσμα, προκύπτει ένα trade-off στην προσέγγισή μας. Πολλαπλασιάζοντας τις πιθανότητες με μικρές δυνάμεις του 10 μειώνεται σημαντικά η ακρίβεια του Naive Bayes, καθώς οι πιθανότητες δεν απεικονίζονται ικανοποιητικά. Αντίθετα, πολλαπλασιασμός των πιθανοτήτων με πολύ μεγάλες δυνάμεις του 10 ενδέχεται να οδηγήσει σε κακή απόδοση, καθώς μπορεί να υπάρξουν υπερχειλίσεις στους buffers του προγράμματος eBPF, όταν πραγματοποιούνται οι απαιτούμενοι πολλαπλασιασμοί των πιθανοτήτων του Naive Bayes.

Βασικό μέρος της αξιολόγησης του μηχανισμού αποτελεί η εκτίμηση του παραπάνω trade-off. Συγκεκριμένα, ελέγχουμε πώς η απεικόνιση των πιθανοτήτων του Naive Bayes στο επίπεδο δεδομένων επηρεάζει την ακρίβεια του αλγορίθμου να ταξινομεί τα εισερχόμενα ερωτήματα DNS ως καλόβουλα ή κακόβουλα. Πολλαπλασιάζοντας τις πιθανότητες του Naive Bayes με διάφορες δυνάμεις του 10 παρατηρούμε ότι ούτε μικρές, αλλά ούτε και μεγάλες δυνάμεις εξασφαλίζουν καλή ακρίβεια για το μοντέλο. Αντίθετα, ενδιάμεσες τιμές πετυχαίνουν ικανοποιητική ακρίβεια ταξινόμησης ερωτημάτων DNS (περίπου 99%).

Επιπρόσθετα, αξιολογούμε τις δυνατότητες του μηχανισμού μας να αντιμετωπίζει επιθέσεις Water Torture, όταν εγκαθίσταται σε Recursors. Προωθώντας ταυτόχρονα καλόβουλα και κακόβουλα ερωτήματα DNS σε έναν Recursor παρατηρούμε ότι ο προτεινόμενος data plane μηχανισμός προστατεύει αποτελεσματικά τον Recursor, φιλτράροντας το μεγαλύτερο μέρος της κακόβουλης κίνησης.

Η υλοποίηση αλγορίθμων μηχανικής μάθησης στο επίπεδο δεδομένων αποδεικνύεται πολλά υποσχόμενη τεχνική για την αντιμετώπιση επιθέσεων DDoS. Ωστόσο, αν και η μηχανική μάθηση έχει χρησιμοποιεί αρκετά σε ερευνητικές εργασίες για την ανάπτυξη μηχανισμών άμυνας, δεν έχει βρει ακόμα ευρεία εφαρμογή σε παραγωγικά περιβάλλοντα. Το πρόβλημα είναι, συνήθως, ότι οι διαχειριστές δικτύων είναι απρόθυμοι να χρησιμοποιήσουν μοντέλα, των οποίων τις αποφάσεις δεν κατανοούν πλήρως. Στο επόμενο κεφάλαιο θα χρησιμοποιήσουμε τεχνικές επεξηγήσιμης τεχνητής νοημοσύνης (XAI) για να ερμηνεύσουμε τις αποφάσεις αλγορίθμων μηχανικής μάθησης που χρησιμοποιούνται για την ταξινόμηση ερωτημάτων DNS που παράγονται από DGA's.

10.7 Κεφάλαιο 7

Στο συγκεκριμένο κεφάλαιο αξιοποιούμε τεχνικές επεξηγήσιμης τεχνητής νοημοσύνης (eXplainable Artificial Intelligence, XAI) για να ερμηνεύσουμε τις αποφάσεις ταξινομητών

DNS. Συγκεκριμένα, μελετάμε τη λειτουργία δυαδικών ταξινομητών που διαχωρίζουν καλόβουλα ονόματα από κακόβουλα που παράγονται από DGA's. Χρησιμοποιώντας αλγορίθμους XAI αποσκοπούμε στη διευκόλυνση της υιοθέτησης των παραπάνω ταξινομητών σε δικτυακές υποδομές, όπου απαιτούνται αιτιολογήσεις για τα κριτήρια απόφασης black-box μοντέλων μηχανικής μάθησης.

Η προσέγγισή μας βασίζεται στη μέθοδο SHapley Additive exPlanation (SHAP) λόγω των πλεονεκτημάτων που προσφέρει σε σχέση με άλλες τεχνικές XAI. Σημαντικό πλεονέκτημα της SHAP είναι ότι μπορεί να εξάγει καθολικές (global), αλλά και τοπικές (local) ερμηνείες μοντέλων μηχανικής μάθησης. Οι καθολικές ερμηνείες παρέχονται για πολυάριθμα δειγματικά σημεία του συνόλου δεδομένων, ενώ οι τοπικές ερμηνείες αφορούν μεμονωμένα δειγματικά σημεία. Παράλληλα, ένα ακόμη σημαντικό πλεονέκτημα της SHAP είναι ότι λειτουργεί με model-agnostic, post-hoc τρόπο. Έτσι, μπορεί να εφαρμοστεί για οποιονδήποτε αλγόριθμο μηχανικής μάθησης (model-agnostic), αφού ολοκληρωθεί η διαδικασίας εκπαίδευσης (post-hoc). Η προσέγγισή μας στηρίζεται στις παραπάνω ιδιότητες της SHAP για να ερμηνεύσει τις αποφάσεις ταξινομητών που βασίζονται σε δέντρα αποφάσεων και σε βαθιά νευρωνικά δίκτυα (deep neural networks) με ενιαίο τρόπο.

Η προσέγγισή μας περιλαμβάνει 50 features, τα οποία εξάγονται εξ ολοκλήρου από τα ονόματα, αποδίδοντας τις στατιστικές και γλωσσικές ιδιότητές τους. Κάποια ενδεικτικά features είναι το μήκος του domain name prefix, ο αριθμός των φωνηέντων/συμφώνων στο prefix, η συχνότητα δεκαδικών ψηφίων και γραμμάτων του λατινικού αλφαβήτου, το πλήθος των υπακολουθιών του prefix που αντιστοιχούν σε υπαρκτές λέξεις, καθώς και η εντροπία του prefix. Features που βασίζονται σε ιστορικές πληροφορίες αποφεύγονται. Τέτοια features απαιτούν, συνήθως, χρονοβόρες προσβάσεις σε εξωτερικές βάσεις δεδομένων, ενώ ενδέχεται να εκθέσουν προσωπικές πληροφορίες των χρηστών ενός δικτύου.

Η αξιολόγηση εστιάζει στην αποτίμηση της συνεισφοράς των features στις εξόδους των επιλεγμένων μοντέλων μηχανικής μάθησης, τον έλεγχο της επιρροής συγκεκριμένων τιμών στις αποφάσεις των ταξινομητών, καθώς και στις αλληλεπιδράσεις των features. Η ανάλυσή μας βασίζεται σε πολυάριθμα εργαλεία οπτικοποίησης που παρέχει η SHAP και συγκεκριμένα στα summary, dependence και force plots. Τα summary plots απεικονίζουν τα features σε φθίνουσα ταξινόμηση ανάλογα με την επίδρασή τους στις αποφάσεις του μοντέλου, ενώ παράλληλα δείχνουν πώς μικρές/μεγάλες τιμές των features επηρεάζουν τις ταξινομήσεις. Τα dependence plots εστιάζουν σε συγκεκριμένα features του αλγορίθμου και τις επιρροές των τιμών τους στις αποφάσεις του μοντέλου, ενώ καθιστούν δυνατή τη μελέτη αλληλεξαρτήσεων ανάμεσα στο εξεταζόμενο feature και τα υπόλοιπα. Τέλος, τα force plots απεικονίζουν τη συνεισφορά των features για μεμονωμένα δειγματικά σημεία του συνόλου δεδομένων, δηλαδή σχετίζονται με τοπικές ερμηνείες.

Τα πειράματά μας περιλαμβάνουν την εκπαίδευση και αξιολόγηση πλήθους δυαδικών ταξινομητών για την κατηγοριοποίηση ονομάτων που παράγονται από DGA's. Αρχικά, μελετάμε ταξινομητές που βασίζονται σε δέντρα αποφάσεων. Συγκεκριμένα, οι αλγόριθμοι δέντρων που συμπεριλάβαμε στην αξιολόγησή μας ήταν οι Random Forests, Gradient

Boosting, eXtreme Gradient Boosting (XGBoost), Adaptive Boosting (AdaBoost) και Extremely Randomized Trees (ExtraTrees). Από τους αλγορίθμους αυτούς, η μεγαλύτερη ακρίβεια σημειώθηκε από τον XGBoost. Έπειτα, τα πειράματά μας εστίασαν σε βαθιά νευρωνικά δίκτυα και, συγκεκριμένα, σε Multi-Layer Perceptrons (MLP's).

Στη συνέχεια λήφθηκαν ερμηνείες για τη λειτουργία του αλγορίθμου XGBoost (καλύτερος δενδρικός ταξινομητής) και για το βαθύ νευρωνικό δίκτυο, δηλαδή το MLP. Ιδιαίτερη έμφαση στην αξιολόγησή μας δόθηκε στη σύγκριση του τρόπου απόφασης των παραπάνω αλγορίθμων, καθώς και στο πώς οι αποφάσεις των ταξινομητών επηρεάζονται από τη μέθοδο που χρησιμοποιείται για την παραγωγή ονομάτων DGA. Οι μέθοδοι παραγωγής που εξετάσαμε περιελάμβαναν όλες τις ήδη υπάρχουσες, δηλαδή arithmetic, wordlist, hash και permutation based.

Οι arithmetic-based DGA's κατασκευάζουν ονόματα ενώνοντας χαρακτήρες με τυχαίο τρόπο. Έτσι, τα ονόματα που παράγονται από arithmetic-based DGA's χαρακτηρίζονται συνήθως από μεγάλες ακολουθίες συμφώνων και αυξημένη τυχαιότητα σε σχέση με τα καλόβουλα ονόματα. Οι wordlist-based DGA's παράγουν ονόματα ενώνοντας λέξεις που επιλέγονται τυχαία από κάποιο διαθέσιμο λεξικό. Επομένως, τα ονόματα που παράγονται από wordlist-based DGA's μοιάζουν σε μεγάλο βαθμό με καλόβουλα ονόματα με αποτέλεσμα να δυσκολεύει σημαντικά η ανίχνευσή τους. Οι hash-based DGA's βασίζονται στα αποτελέσματα συναρτήσεων κατακερματισμού και κατασκευάζουν ονόματα χρησιμοποιώντας τις δεκαεξαδικές αναπαραστάσεις τους. Ως αποτέλεσμα, τα ονόματα που παράγονται από hash-based DGA's χαρακτηρίζονται από υψηλές συχνότητες εμφάνισης δεκαεξαδικών ψηφίων (ψηφία 0-9 και χαρακτήρες A-F). Οι permutation-based DGA's στηρίζονται στις πολυάριθμες πιθανές μεταθέσεις χαρακτήρων ενός αλφαριθμητικού και τα όνοματα που παράγονται από αυτούς χαρακτηρίζονται από τις στατιστικές και γλωσσικές ιδιότητες του αρχικού αλφαριθμητικού.

Η εκπαίδευση, αξιολόγηση και ερμηνεία των παραπάνω μοντέλων μηχανικής μάθησης βασίστηκε σε σύγχρονα σύνολα δεδομένων που χρησιμοποιούνται κατά κόρον σε ταξινομητές κίνησης που παράγεται από DGA's. Συγκεκριμένα, ως καλόβουλα ονόματα χρησιμοποιήσαμε εγγραφές από τη λίστα Tranco, ενώ τα κακόβουλα ονόματα επιλέχθηκαν από το αποθετήριο DGArchive.

10.8 Κεφάλαιο 8

Στο κεφάλαιο αυτό συνοψίζουμε τα βασικά συμπεράσματα της διατριβής και περιγράφουμε πιθανές μελλοντικές κατευθύνσεις για έρευνα. Μερικές από τις κατευθύνσεις αυτές είναι:

- Η επέκταση των προτεινόμενων μηχανισμών για την ανίχνευση και αντιμετώπιση επιπρόσθετων επιθέσεων DNS, όπως είναι οι DNS Amplification και NXNSAttack.
- Η σύγκριση της απόδοσης των μηχανισμών που αναπτύχθηκαν σε XDP με μηχανισμούς που βασίζονται σε άλλες μεθόδους προγραμματισμού στο επίπεδο δεδομένων,

π.χ. P4 και DPDK.

- Η διερεύνηση νέων πιθανοτικών δομών δεδομένων, π.χ. τα φίλτρα Morton και τα φίλτρα Vacuum, για την αντικατάσταση των BF's και CF's στους μηχανισμούς μας.
- Η χρήση ομόσπονδης μάθησης (Federated Learning) για τη συνεργατική ανίχνευση δικτυακών επιθέσεων με τρόπο που σέβεται την ιδιωτικότητα των συμμετεχόντων.
- Εφαρμογή τεχνικών XAI για την ερμηνεία των αποφάσεων αλγορίθμων μη επιβλεπόμενης μηχανικής μάθησης.

10.9 Κεφάλαιο 9

Στο συγκεκριμένο κεφάλαιο παρουσιάζονται οι ερευνητικές εργασίες που πραγματοποιήθηκαν σε περιοδικά, συνέδρια και workshops.

Αντιστοιχία Αγγλικών-Ελληνικών Όρων

Application layer Επίπεδο εφαρμογής

Authoritative DNS Server Επίσημος εξυπηρετητής DNS

Big Data Μεγάλα δεδομένα

Bloom Filter Φίλτρο Bloom

Botnet Δίκτυο μολυσμένων συσκευών

Bot Μολυσμένη συσκευή

Brute-force attack Επίθεση ωμής βίας

Cloud infrastructure Υποδομή συννέφου

Cloud service Υπηρεσία συννέφου

Command & Control (C&C) server Εξυπηρετητής ελέγχου

Component Δομικό στοιχείο

Cuckoo Filter (CF) Φίλτρο Cuckoo

Data plane programming Προγραμματισμός στο επίπεδο δεδομένων

Data plane Επίπεδο δεδομένων

Dataset Σύνολο δεδομένων

Deep neural network Βαθύ νευρωνικό δίκτυο

Deep Packet Inspection (DPI) Βαθιά επιθεώρηση πακέτου

Deterministic Αιτιοκρατικός

Distributed Denial of Service (DDoS) Κατανεμημένη άρνηση παροχής υπηρεσίας

DNS cache Προσωρινή μνήμη DNS

Domain Generation Algorithm (DGA) Αλγόριθμος παραγωγής ονομάτων

Domain Name System (DNS) Σύστημα Ονοματοδοσίας

Domain name 'Ονομα

Drivers Οδηγοί

eBPF program Πρόγραμμα eBPF

eXplainable Artificial Intelligence (XAI) Επεξηγήσιμη τεχνητή νοημοσύνη

False Negative (FN) Ψευδώς αρνητικό

False Positive (FP) Ψευδώς θετικό

Feature Χαρακτηριστικό

Federated Learning Ομόσπονδη μάθηση

Firewall Τείχος προστασίας

Frequency estimation Εκτίμηση συχνότητας

Global interpretation Καθολική ερμηνεία

Hardware Υλικό

Hash function Συνάρτηση κατακερματισμού

Hashed DNS Zone (HsZn) Hashed ζώνη DNS

Incremental DNS Zone (IncZn) Incremental ζώνη DNS

Internet Protocol (IP) Πρωτόκολλο Διαδικτύου

IP spoofing Παραποίηση διευθύνσεων IP

Linux kernel Πυρήνας Linux

Local interpretation Τοπική ερμηνεία

Machine Learning (ML) Μηχανική μάθηση

Name lookup 'Ελεγχος ονόματος

Natural Language Processing (NLP) Επεξεργασία φυσικής γλώσσας

Payload Δεδομένα

Plaintext DNS Zone (PltZn) Plaintext ζώνη DNS

Prefix Πρόθεμα

Privacy Ιδιωτικότητα

Probabilistic data structure Πιθανοτική δομή δεδομένων

Recursive DNS Server (Recursor) Αναδρομικός εξυπηρετητής DNS

Software-Defined Network (SDN) Δίκτυο που καθορίζεται από λογισμικό

Third-party Εξωτερικός συνεργάτης

True Positive (TP) Αληθώς θετικό

User space Χώρος χρήστη

Whitelist Λίστα καλόβουλων ονομάτων

Bibliography

[1] Paul Mockapetris, *Domain Names - Implementation and Specification*, RFC 1035, November 1987. Available at: https://datatracker.ietf.org/doc/html/rfc1035 [Accessed February 2024].

[2] Romain Fouchereau, *Cyber Threat Intelligence - IDC 2023 Global DNS Threat Report*, EfficientIP, August 2023. Available at: https://efficientip.com/resources/cyber-threat-intelligence-idc-2023-global-dns-threat-report/ [Accessed February 2024].

[3] James Scott and Drew Spaniel, *Rise of the Machines: The Dyn Attack Was Just a Practice Run*, Institute for Critical Infrastructure Technology, December 2016. Available at: https://media.kasperskycontenthub.com/wp-content/uploads/sites/62/2017/02/21140649/ICIT-Brief-Rise-of-the-Machines.pdf [Accessed February 2024].

[4] John Wagnon, *Lightboard Lessons: The DNS Water Torture Attack*, DevCentral (F5), October 2018. Available at: https://community.f5.com/t5/technical-articles/lightboard-lessons-the-dns-water-torture-attack/ta-p/281711 [Accessed February 2024].

[5] Yuya Takeuchi, Takuro Yoshida, Ryotaro Kobayashi, Masahiko Kato, and Hiroyuki Kishimoto, *Detection of the DNS Water Torture Attack by Analyzing Features of the Subdomain Name*, in Journal of Information Processing, Volume 24, Issue 5, pp. 793–801, September 2016.

[6] Daniel Plohmann, Khaled Yakdan, Michael Klatt, Johannes Bader, and Elmar Gerhards-Padilla, *A Comprehensive Measurement Study of Domain Generating Malware*, in the Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, August 2016, pp. 263–278.

[7] *Domain Count Statistics for TLD's*, DomainTools. Available at: https://research.domaintools.com/statistics/tld-counts/ [Accessed February 2024].

[8] Marcin Skwarek, Maciej Korczyński, Wojciech Mazurczyk, and Andrzej Duda, *Characterizing Vulnerability of DNS AXFR Transfers with Global-Scale Scanning*, in the Proceedings of the 2019 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, May 2019, pp. 193–198.

[9] Andrii Gakhov, *Probabilistic Data Structures and Algorithms for Big Data Applications*, Published by Books on Demand, August 2022.

[10] Toke Høiland-Jørgensen, Jesper Dangaard Brouer, Daniel Borkmann, John Fastabend, Tom Herbert, David Ahern, and David Miller, *The eXpress Data Path: Fast Programmable Packet Processing in the Operating System Kernel*, in the Proceedings of the 14th ACM International Conference on Emerging Networking Experiments and Technologies (ACM CoNEXT 2018), Heraklion, Greece, December 2018, pp. 54–66.

[11] *DPDK - Powering the Future of Network Software and Applications*, The Linux Foundation. Available at: https://www.dpdk.org/ [Accessed February 2024].

[12] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, and David Walker, *P4: Programming Protocol-Independent Packet Processors*, in ACM SIGCOMM Computer Communication Review, Volume 44, Issue 3, pp. 87–95, July 2014.

[13] Christoph Molnar, *Interpretable Machine Learning - A Guide for Making Black Box Models Explainable*, February 2022. Available at: https://christophm.github.io/interpretable-ml-book/ [Accessed February 2024].

[14] Leonida Gianfagna and Antonio Di Cecco, *Explainable AI with Python*, Published by Springer, April 2021.

[15] Scott M. Lundberg and Su-In Lee, *A Unified Approach to Interpreting Model Predictions*, in the Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, December 2017, pp. 4768–4777.

[16] Burton H. Bloom, *Space/Time Trade-offs in Hash Coding with Allowable Errors*, in Communications of the ACM, Volume 13, Issue 7, pp. 422–426, July 1970.

[17] Graham Cormode and Shan Muthukrishnan, *An Improved Data Stream Summary: The Count-Min Sketch and Its Applications*, in Elsevier Journal of Algorithms, Volume 55, Issue 1, pp. 58–75, April 2005.

[18] *SymSpell: Spelling Correction and Fuzzy Search*, GitHub repository of Wolf Garbe. Available at: https://github.com/wolfgarbe/SymSpell [Accessed February 2024].

[19] Bin Fan, David G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher, *Cuckoo Filter: Practically Better Than Bloom*, in the Proceedings of the 10th ACM International Conference on Emerging Networking Experiments and Technologies (ACM CoNEXT 2014), Sydney, Australia, December 2014, pp. 75–88.

[20] *DGArchive Repository Access Portal*, Cyber Analysis and Defense Department of Fraunhofer FKIE. Available at: https://dgarchive.caad.fkie.fraunhofer.de/welcome/ [Accessed February 2024].

[21] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen, *Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation*, in arXiv Preprint arXiv:1806.01156, December 2018. Available at: https://arxiv.org/pdf/1806.01156.pdf [Accessed February 2024].

[22] José Jair Santanna, Roland van Rijswijk-Deij, Rick Hofstede, Anna Sperotto, Mark Wierbosch, Lisandro Zambenedetti Granville, and Aiko Pras, *Booters - An Analysis of DDoS-as-a-Service Attacks*, in the Proceedings of the 14th IFIP/IEEE International Symposium on Integrated Network Management (IFIP/IEEE IM 2015), Ottawa, Canada, May 2015, pp. 243–251.

[23] James Kurose and Keith Ross, *Computer Networking: A Top-Down Approach*, 6th Edition, Published by Pearson, March 2012.

[24] Celso Martinho and Tom Strickx, *Understanding how Facebook Disappeared from the Internet*, The Cloudflare Blog, May 2021. Available at: https://blog.cloudflare.com/october-2021-facebook-outage/ [Accessed February 2024].

[25] *BIND 9 - Versatile, Classic, Complete Name Server Software*, Internet Systems Consortium. Available at: https://www.isc.org/bind/ [Accessed February 2024].

[26] *PowerDNS - A Faster, Safer, and More Secure Internet*, PowerDNS. Available at: https://www.powerdns.com/ [Accessed February 2024].

[27] Sandeep Sarat, Vasileios Pappas, and Andreas Terzis, *On the Use of Anycast in DNS*, in the Proceedings of the 2005 ACM International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS 2005), Banff, Alberta, Canada, June 2005, pp. 394–395.

[28] Marc Kührer, Thomas Hupperich, Jonas Bushart, Christian Rossow, and Thorsten Holz, *Going Wild: Large-Scale Classification of Open DNS Resolvers*, in the Proceedings of the 2015 ACM Internet Measurement Conference (ACM IMC '15), Tokyo, Japan, October 2015, pp. 355–368.

[29] Giovane C. M. Moura, John Heidemann, Ricardo de O. Schmidt, and Wes Hardaker, *Cache Me If You Can: Effects of DNS Time-to-Live*, in the Proceedings of the 2019 ACM Internet Measurement Conference (ACM IMC '19), Amsterdam, The Netherlands, October 2019, pp. 101–115.

[30] Edward P. Lewis and Alfred Hoenes, *DNS Zone Transfer Protocol (AXFR)*, RFC 5936, June 2010. Available at: https://datatracker.ietf.org/doc/html/rfc5936 [Accessed February 2024].

[31] Masataka Ohta, *Incremental Zone Transfer in DNS*, RFC 1995, August 1996. Available at: https://datatracker.ietf.org/doc/html/rfc1995 [Accessed February 2024].

[32] Donald Eastlake, Eric Brunner-Williams, and Bill Manning, *Domain Name System (DNS) IANA Considerations*, RFC 2929, September 2000. Available at: https://datatracker.ietf.org/doc/html/rfc2929 [Accessed February 2024].

[33] Saman Taghavi Zargar, James Joshi, and David Tipper, *A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks*, in IEEE Communications Surveys and Tutorials, Volume 15, Issue 4, pp. 2046–2069, Fourth Quarter 2013.

[34] Zev Brodsky, *The Psychology Behind DDoS: Motivations and Methods*, Perimeter 81, February 2020. Available at: https://www.perimeter81.com/blog/network/the-psychology-behind-ddos-attacks [Accessed February 2024].

[35] Toby Ehrenkranz and Jun Li, *On the State of IP Spoofing Defense*, in ACM Transactions on Internet Technology (TOIT), Volume 9, Issue 2, pp. 1–29, May 2009.

[36] Sérgio S. C. Silva, Rodrigo M. P. Silva, Raquel C. G. Pinto, and Ronaldo M. Salles, *Botnets: A Survey*, in Elsevier Computer Networks, Volume 57, Issue 2, pp. 378–403, October 2012.

[37] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon, *Peer-to-Peer Botnets: Overview and Case Study*, in the Proceedings of the 1st USENIX Workshop on Hot Topics in Understanding Botnets (HotBots 2007), Cambridge, MA, USA, April 2007.

[38] *Data Center Security*, Wikipedia Article. Available at: https://en.m.wikipedia.org/wiki/Data_center_security [Accessed February 2024].

[39] *What is a DDoS attack?*, Cloudflare Website. Available at: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-attack/ [Accessed February 2024].

[40] Marinos Dimolianis, Adam Pavlidis, Dimitris Kalogeras, and Vasilis Maglaris, *Mitigation of Multi-vector Network Attacks via Orchestration of Distributed Rule Placement*, in the Proceedings of the 16th IFIP/IEEE International Symposium on Integrated Network and Service Management (IFIP/IEEE IM 2019), Washington, DC, USA, April 2019, pp. 162–170.

[41] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung, *Intrusion Detection System: A Comprehensive Review*, in Elsevier Journal of Network and Computer Applications, Volume 36, Issue 1, pp. 16–24, September 2012.

[42] Kazeem B. Adedeji, Adnan M. Abu-Mahfouz, and Anish M. Kurien, *DDoS Attack and Detection Methods in Internet-Enabled Networks: Concept, Research Perspectives, and*

Challenges, in MDPI Journal of Sensor and Actuator Networks, Volume 12, Issue 4, p. 51, July 2023.

[43] Karanbir Singh, Kanwalvir Singh Dhindsa, and Deepa Nehra, *T-CAD: A Threshold Based Collaborative DDoS Attack Detection in Multiple Autonomous Systems*, in Elsevier Journal of Information Security and Applications, Volume 51, p. 102457, February 2020.

[44] Jisa David and Ciza Thomas, *Efficient DDoS Flood Attack Detection Using Dynamic Thresholding on Flow-based Network Traffic*, in Elsevier Computers and Security, Volume 82, pp. 284–295, January 2019.

[45] Wu Zhijun, Li Wenjing, Liu Liang, and Yue Meng, *Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey*, in IEEE Access, Volume 8, pp. 43920–43943, March 2020.

[46] Marinos Dimolianis, Adam Pavlidis, and Vasilis Maglaris, *Signature-Based Traffic Classification and Mitigation for DDoS Attacks Using Programmable Network Data Planes*, in IEEE Access, Volume 9, pp. 113061–113076, August 2021.

[47] Juma Ibrahim and Slavko Gajin, *Entropy-based Network Traffic Anomaly Classification Method Resilient to Deception*, in Computer Science and Information Systems, Volume 19, Issue 1, pp. 87–116, January 2022.

[48] Matthias Wichtlhuber, Eric Strehle, Daniel Kopp, Lars Prepens, Stefan Stegmueller, Alina Rubina, Christoph Dietzel, and Oliver Hohlfeld, *IXP Scrubber: Learning from Blackholing Traffic for ML-Driven DDoS Detection at Scale*, in the Proceedings of the ACM SIGCOMM 2022 Conference (ACM SIGCOMM '22), Amsterdam, The Netherlands, August 2022, pp. 707–722.

[49] Narmeen Zakaria Bawany, Jawwad A. Shamsi, and Khaled Salah, *DDoS Attack Detection and Mitigation Using SDN: Methods, Practices, and Solutions*, in Arabian Journal for Science and Engineering, Volume 42, pp. 425–441, February 2017.

[50] Pedro R. Marques, Nischal Sheth, Robert Raszuk, Barry Greene, Jared Mauch, and Danny R. McPherson, *Dissemination of Flow Specification Rules*, RFC 5575, August 2009. Available at: https://datatracker.ietf.org/doc/html/rfc5575 [Accessed February 2024].

[51] *DNS Flood*, Imperva. Available at: https://www.imperva.com/learn/ddos/dns-flood/ [Accessed February 2024].

[52] Xi Luo, Liming Wang, Zhen Xu, Kai Chen, Jing Yang, and Tian Tian, *A Large Scale Analysis of DNS Water Torture Attack*, in the Proceedings of the 2nd ACM International Conference on Computer Science and Artificial Intelligence (ACM CSAI '18), Shenzhen, China, December 2018, pp. 168–173).

[53] Takuro Yoshida, Kento Kawakami, Ryotaro Kobayashi, Masahiko Kato, Masayuki Okada, and Hiroyuki Kishimoto, *Detection and Filtering System for DNS Water Torture Attacks Relying Only on Domain Name Information*, in Journal of Information Processing, Volume 25, pp. 854–865, September 2017.

[54] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis, *DNS Amplification Attack Revisited*, in Elsevier Computers and Security, Volume 39, pp. 475–485, October 2013.

[55] Roy Arends, Rob Austein, Matt Larson, Dan Massey, and Scott Rose, *DNS Security Introduction and Requirements*, RFC 4033, March 2005. Available at: https://datatracker.ietf.org/doc/html/rfc4033 [Accessed February 2024].

[56] Paul Hoffman, *DNS Security Extensions (DNSSEC)*, RFC 9364, February 2023. Available at: https://datatracker.ietf.org/doc/html/rfc9364 [Accessed February 2024].

[57] *DNS Amplification*, Imperva. Available at: https://www.imperva.com/learn/ddos/dns-amplification/ [Accessed February 2024].

[58] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja, and Daniel Grant, *Predicting Domain Generation Algorithms with Long Short-Term Memory Networks*, in arXiv Preprint arXiv:1611.00791, November 2016. Available at: https://arxiv.org/pdf/1611.00791.pdf [Accessed February 2024].

[59] Bin Yu, Jie Pan, Daniel Gray, Jiaming Hu, Chhaya Choudhary, Anderson C. A. Nascimento, and Martine De Cock, *Weakly Supervised Deep Learning for the Detection of Domain Generation Algorithms*, in IEEE Access, Volume 7, pp. 51542–51556, April 2019.

[60] Juhong Namgung, Siwoon Son, and Yang-Sae Moon, *Efficient Deep Learning Models for DGA Domain Detection*, in Hindawi Security and Communication Networks, Volume 2021, pp. 1–15, January 2021.

[61] Kyung Ho Park, Hyun Min Song, Jeong Do Yoo, Su-Youn Hong, Byoungmo Cho, Kwangsoo Kim, and Huy Kang Kim, *Unsupervised Malicious Domain Detection with Less Labeling Effort*, in Elsevier Computers and Security, Volume 116, p. 102662, February 2022.

[62] Bin Yu, Jie Pan, Jiaming Hu, Anderson Nascimento, and Martine De Cock, *Character Level Based Detection of DGA Domain Names*, in the Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, July 2018, pp. 1–8.

[63] Luhui Yang, Guangjie Liu, Yuewei Dai, Jinwei Wang, and Jiangtao Zhai, *Detecting Stealthy Domain Generation Algorithms Using Heterogeneous Deep Neural Network Framework*, in IEEE Access, Volume 8, pp. 82876–-82889, April 2020.

[64] Samuel Schüppen, Dominik Teubert, Patrick Herrmann, and Ulrike Meyer, *FANCI: Feature-based Automated NXDomain Classification and Intelligence*, in the Proceedings of the 27th USENIX Security Symposium (USENIX Security 18), Baltimore, MD, USA, August 2018, pp. 1165–1181.

[65] Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon, *From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware*, in the Proceedings of the 21st USENIX Security Symposium (USENIX Security 12), Bellevue, WA, USA, August 2012, pp. 491–-506.

[66] Leyla Bilge, Sevil Sen, Davide Balzarotti, Engin Kirda, and Christopher Kruegel, *EXPOSURE: A Passive DNS Analysis Service to Detect and Report Malicious Domains*, in ACM Transactions on Information and System Security (TISSEC), Volume 16, Issue 4, pp. 1–28, April 2014.

[67] Ahmad O. Almashhadani, Mustafa Kaiiali, Domhnall Carlin, and Sakir Sezer, *MaldomDetector: A System for Detecting Algorithmically Generated Domain Names with Machine Learning*, in Elsevier Computers and Security, Volume 93, p. 101787, March 2020.

[68] Claudio Marques, Silvestre Malta, and João Magalhães, *DNS Firewall Based on Machine Learning*, in MDPI Future Internet, Volume 13, Issue 12, p. 309, November 2021.

[69] Diego Kreutz, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig, *Software-Defined Networking: A Comprehensive Survey*, in the Proceedings of the IEEE, Volume 103, Issue 1, pp. 14–76, December 2014.

[70] Wenfeng Xia, Yonggang Wen, Chuan Heng Foh, Dusit Niyato, and Haiyong Xie, *A Survey on Software-Defined Networking*, in IEEE Communications Surveys and Tutorials, Volume 17, Issue 1, pp. 27–51, First Quarter 2015.

[71] Fei Hu, Qi Hao, and Ke Bao, *A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation*, in IEEE Communications Surveys and Tutorials, Volume 16, Issue 4, pp. 2181–2206, Fourth Quarter 2014.

[72] Bhargavi Goswami, Shuwen Hu, and Yanming Feng, *Software-Defined Networking for Real-Time Network Systems*, in the Springer Handbook of Real-Time Computing, pp. 935–959, 2022.

[73] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner, *OpenFlow: Enabling Innovation in Campus Networks*, in ACM SIGCOMM Computer Communication Review, Volume 38, Issue 2, pp. 69–74, April 2008.

[74] Seyed Mohammad Mousavi and Marc St-Hilaire, *Early Detection of DDoS Attacks Against SDN Controllers*, in the Proceedings of the 2015 IEEE International Conference on Computing, Networking and Communications (IEEE ICNC 2015), Anaheim, CA, USA, February 2015, pp. 77–81.

[75] Open Networking Foundation, *OpenFlow Switch Specification*, March 2015. Available at: https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf [Accessed February 2024].

[76] Oliver Michel, Roberto Bifulco, Gábor Rétvári, and Stefan Schmid, *The Programmable Data Plane: Abstractions, Architectures, Algorithms, and Applications*, in ACM Computing Surveys, Volume 54, Issue 4, pp. 1–36, April 2021.

[77] Frederik Hauser, Marco Häberle, Daniel Merling, Steffen Lindner, Vladimir Gurevich, Florian Zeiger, Reinhard Frank, and Michael Menth, *A Survey on Data Plane Programming with P4: Fundamentals, Advances, and Applied Research*, in Elsevier Journal of Network and Computer Applications, Volume 212, p. 103561, December 2022.

[78] The P4 Language Consortium, *P4-16 Language Specification*, May 2017. Available at: https://p4.org/p4-spec/docs/P4-16-v1.0.0-spec.html [Accessed February 2024].

[79] *Behavioral Model (bmv2)*, GitHub Repository "p4lang". Available at: https://github.com/p4lang/behavioral-model [Accessed February 2024].

[80] *Netronome Agilio CX Dual-Port 25 Gigabit Ethernet SmartNIC*. Available at: https://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3144 [Accessed February 2024].

[81] *NetFPGA-SUME FPGA Development Board*. Available at: https://www.xilinx.com/products/boards-and-kits/1-6ogkf5.html [Accessed February 2024].

[82] *Edgecore Wedge 100BF-32X 32-Port 100GbE Bare Metal Switch*. Available at: http://www.colfaxdirect.com/store/pc/viewPrd.asp?idproduct=3485 [Accessed February 2024].

[83] Changgang Zheng, Mingyuan Zang, Xinpeng Hong, Riyad Bensoussane, Shay Vargaftik, Yaniv Ben-Itzhak, and Noa Zilberman, *Automating In-Network Machine Learning*, in arXiv Preprint arXiv:2205.08824, May 2022. Available at: https://arxiv.org/pdf/2205.08824.pdf [Accessed February 2024].

[84] Danilo Cerović, Valentin Del Piccolo, Ahmed Amamou, Kamel Haddadou, and Guy Pujolle, *Fast Packet Processing: A Survey*, in IEEE Communications Surveys and Tutorials, Volume 20, Issue 4, pp. 3645–3676, Fourth Quarter 2018.

[85] *What is DPDK?*, Packet Coders. Available at: https://www.packetcoders.io/what-is-dpdk/ [Accessed February 2024].

[86] Benoît Claise, *Cisco Systems NetFlow Services Export Version 9*, RFC 3954, October 2004. Available at: https://datatracker.ietf.org/doc/html/rfc3954 [Accessed February 2024].

[87] *NetFlow*, Wikipedia Article. Available at: https://en.wikipedia.org/wiki/NetFlow [Accessed February 2024].

[88] Peter Phaal, Sonia Panchen, and Neil McKee, *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*, RFC 3176, September 2001. Available at: https://datatracker.ietf.org/doc/html/rfc3176 [Accessed February 2024].

[89] *collectd – The System Statistics Collection Daemon*. Available at: https://collectd.org/ [Accessed February 2024].

[90] *RRDtool - Round Robin Database Tool*, GitHub Repository of Tobias Oetiker. Available at: https://github.com/oetiker/rrdtool-1.x [Accessed February 2024].

[91] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz, *Theory and Practice of Bloom Filters for Distributed Systems*, in IEEE Communications Surveys and Tutorials, Volume 14, Issue 1, pp. 131–155, First Quarter 2012.

[92] Li Fan, Pei Cao, Jussara Almeida, and Andrei Z. Broder, *Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol*, in IEEE/ACM Transactions on Networking, Volume 8, Issue 3, pp. 281–293, June 2000.

[93] Chunchun Zhao and Sartaj Sahni, *String Correction Using the Damerau-Levenshtein Distance*, in BMC Bioinformatics, Volume 20, Issue 11, pp. 1–28, June 2019.

[94] Peter Norvig, *How to Write a Spelling Corrector*, August 2016. Available at: https://norvig.com/spell-correct.html [Accessed February 2024].

[95] Tom Mitchell, *Machine Learning*, Published by McGraw-Hill, January 1997.

[96] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, Published by MIT Press, November 2016. Available at: https://www.deeplearningbook.org/ [Accessed February 2024].

[97] Robert E. Schapire and Yoav Freund, *Boosting: Foundations and Algorithms*, Published by MIT Press, 2012.

[98] Tom Mitchell, *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression*. Available at: https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf [Accessed February 2024].

[99] *GDPR: General Data Protection Regulation*, Intersoft Consulting. Available at: https://gdpr-info.eu/ [Accessed February 2024].

[100] *SHAP KernelExplainer*. Available at: https://shap-lrjball.readthedocs.io/en/latest/generated/shap.KernelExplainer.html [Accessed February 2024].

[101] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou, *Understanding the Mirai Botnet*, in the Proceedings of the 26th USENIX Security Symposium (USENIX Security 17), Vancouver, BC, Canada, August 2017, pp. 1093–1110.

[102] Liron Segal, *Mirai: The IoT Bot that Took Down Krebs and Launched a Tbps Attack on OVH*, F5 Labs, October 2016. Available at: https://www.f5.com/labs/articles/threat-intelligence/mirai-the-iot-bot-that-took-down-krebs-and-launched-a-tbps-attack-on-ovh-22422 [Accessed February 2024].

[103] Negar Mosharraf, Anura P. Jayasumana, and Indrakshi Ray, *A Responsive Defense Mechanism Against DDoS Attacks*, in the Proceedings of the 7th International Symposium on Foundations and Practice of Security (Springer), Montreal, QC, Canada, November 2014, pp. 347–355.

[104] Chenxu Wang, Tony T. N. Miu, Xiapu Luo, and Jinhe Wang, *SkyShield: A Sketch-Based Defense System Against Application Layer DDoS Attacks*, in IEEE Transactions on Information Forensics and Security, Volume 13, Issue 3, pp. 559–573, March 2018.

[105] Changhua Sun, Bin Liu and Lei Shi, *Efficient and Low-Cost Hardware Defense Against DNS Amplification Attacks*, in the Proceedings of the 2008 IEEE Global Telecommunications Conference (IEEE GLOBECOM 2008), New Orleans, LA, USA, November 2008, pp. 1–5.

[106] David Koster, Jason Nikolai, Adam Reznechek, and Andrew Thorstensen, *Recursive Domain Name Service (DNS) Prefetching*, U.S. Patent No. US 10,587,648 B2, Filed April 2017.

[107] ETSI, *Network Functions Virtualisation (NFV); Architectural Framework, ETSI GS NFV 002 V1.2.1*, December 2014. Available at: https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.02.01_60/gs_NFV002v010201p.pdf [Accessed February 2024].

[108] *sflowtool*, GitHub Repository of sflow. Available at: https://github.com/sflow/sflowtool [Accessed February 2024].

[109] *sFlow Sampling Rates*, sFlow Blog, June 2009. Available at: https://blog.sflow.com/2009/06/sampling-rates.html [Accessed February 2024].

[110] *Ryu Component-based Software Defined Networking Framework*. Available at: https://ryu-sdn.org/ [Accessed February 2024].

[111] Kostas Giotis, George Androulidakis, and Vasilis Maglaris, *A Scalable Anomaly Detection and Mitigation Architecture for Legacy Networks via an OpenFlow Middlebox*, in Security and Communication Networks (Wiley), Volume 9, Issue 13, pp. 1958–1970, October 2015.

[112] Adam Pavlidis, Giannis Sotiropoulos, Kostas Giotis, Dimitris Kalogeras, and Vasilis Maglaris, *NFV-compliant Traffic Monitoring and Anomaly Detection based on Dispersed Vantage Points in Shared Network Infrastructures*, in the Proceedings of the 4th IEEE International Conference on Network Softwarization (IEEE NetSoft 2018), Montreal, Canada, June 2018, pp. 197–201.

[113] *Russian DNS Leak*, GitHub Repository of Matthew Bryant. Available at: https://github.com/mandatoryprogrammer/RussiaDNSLeak [Accessed February 2024].

[114] *Zone Files of se*, The Swedish Internet Foundation. Available at: https://internetstiftelsen.se/en/zone-data/ [Accessed February 2024].

[115] Wouter B. de Vries, Roland van Rijswijk-Deij, Pieter-Tjerk de Boer, and Aiko Pras, *Passive Observations of a Large DNS Service: 2.5 Years in the Life of Google*, in IEEE Transactions on Network and Service Management (TNSM), Volume 17, Issue 1, pp. 190–200, March 2020.

[116] *Tcpreplay - Pcap Editing and Replaying Utilities*. Available at: https://tcpreplay.appneta.com/ [Accessed February 2024].

[117] *Open vSwitch - Production Quality, Multilayer Open Virtual Switch*. Available at: https://www.openvswitch.org/ [Accessed February 2024].

[118] *SWITCH DNS Statistics*. Available at: https://www.nic.ch/statistics/dns/ [Accessed February 2024].

[119] *BIND 9 Administrator Reference Manual, page 135*, Internet Systems Consortium. Available at: https://ftp.isc.org/isc/bind9/9.10.8/doc/arm/Bv9ARM.pdf [Accessed February 2024].

[120] *Famous DDoS Attacks: The Largest DDoS Attacks of All Time*, Cloudflare Website. Available at: https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/ [Accessed February 2024].

[121] Gilberto Bertin, *XDP in Practice: Integrating XDP into our DDoS Mitigation Pipeline*, Netdev 2.1, Volume 2, Montreal, Canada, April 2017, pp. 1–5.

[122] Arthur Fabre, *L4Drop: XDP DDoS Mitigations*, The Cloudflare Blog, November 2018. Available at: https://blog.cloudflare.com/l4drop-xdp-ebpf-based-ddos-mitigations/ [Accessed February 2024].

[123] Sam Kottler, *February 28th DDoS Incident Report*, GitHub Blog, March 2018. Available at: https://github.blog/2018-03-01-ddos-incident-report/ [Accessed February 2024].

[124] *Chapter 4 Implementation*, GitHub Repository of Nikos Kostopoulos. Available at: https://www.github.com/nkostopoulos/xdp_dns [Accessed February 2024].

[125] Shir Landau Feibish, Yehuda Afek, Anat Bremler-Barr, Edith Cohen, and Michal Shagam, *Mitigating DNS Random Subdomain DDoS Attacks by Distinct Heavy Hitters Sketches*, in the Proceedings of the 5th ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies (ACM/IEEE HotWeb '17), San Jose, CA, USA, October 2017, pp. 1–6.

[126] Liguo Chen, Yuedong Zhang, Qi Zhao, Guanggang Geng, and ZhiWei Yan, *Detection of DNS DDoS Attacks with Random Forest Algorithm on Spark*, in Elsevier Procedia Computer Science, Volume 134, pp. 310–315, 2018.

[127] Sebastiano Miano, Roberto Doriguzzi-Corin, Fulvio Risso, Domenico Siracusa, and Raffaele Sommese, *Introducing SmartNICs in Server-Based Data Plane Processing: The DDoS Mitigation Use Case*, in IEEE Access, Volume 7, pp. 107161–107170, August 2019.

[128] *A FlowRadar Implementation in eBPF*, GitHub Repository of M. Bentley. Available at: https://github.com/mtbentley/ebpf-flowradar [Accessed February 2024].

[129] Dirk Koelewijn, *Fingerprint-Based Automated Rule Generation for DDoS Mitigation Using the Berkeley Packet Filter*, in the Proceedings of the 30th Twente Student Conference on IT, Enschede, The Netherlands, February 2019.

[130] Oliver Hohlfeld, Johannes Krude, Jens Helge Reelfs, Jan Rüth, and Klaus Wehrle, *Demystifying the Performance of XDP BPF*, in the Proceedings of the 5th IEEE International Conference on Network Softwarization (IEEE NetSoft 2019), Paris, France, June 2019, pp. 208–212.

[131] *Katran Load Balancer*, GitHub Repository of Meta Incubator. Available at: https://github.com/facebookincubator/katran [Accessed February 2024].

[132] *Cilium: eBPF-based Networking, Observability, Security*. Available at: https://cilium.io/ [Accessed February 2024].

[133] Éric Leblond, *Suricata: Why eBPF and XDP in Suricata Matters*, Presentation at SURICON, November 2018. Available at: https://suricon.net/wp-content/uploads/2019/01/SuriCon2018_Leblond.pdf [Accessed February 2024].

[134] *XDPLua*. Available at: https://victornogueirario.github.io/xdplua/ [Accessed February 2024].

[135] Mathieu Corbin, *Introduction to eBPF and XDP*. Available at: https://www.mcorbin.fr/pages/xdp-introduction/ [Accessed February 2024].

[136] *MurmurHash3*, GitHub Repository of Joseph Werle. Available at: https://github.com/jwerle/murmurhash.c [Accessed February 2024].

[137] Nikos Kostopoulos, Adam Pavlidis, Marinos Dimolianis, Dimitris Kalogeras, and Vasilis Maglaris, *A Privacy-Preserving Schema for the Detection and Collaborative Mitigation of DNS Water Torture Attacks in Cloud Infrastructures*, in the Proceedings of the 8th IEEE International Conference on Cloud Networking (IEEE CloudNet 2019), Coimbra, Portugal, November 2019, pp. 1–6.

[138] Valeriy Shevchenko, *DNS Vulnerability for AXFR Queries*, Medium, June 2017. Available at: https://medium.com/@valeriyshevchenko/dns-vulnerability-for-axfr-queries-58a51972fc4d [Accessed February 2024].

[139] *Mitigating DDoS Attacks in Zero Seconds with Proactive Mitigation Controls*, Akamai. Available at: https://www.akamai.com/site/ja/documents/white-paper/proactive-ddos-mitigation-with-prolexic-mitigation-controls-whitepaper.pdf [Accessed February 2024].

[140] *Chapter 5 Implementation*, GitHub Repository of Nikos Kostopoulos. Available at: https://github.com/nkostopoulos/dnspriv [Accessed February 2024].

[141] Steven M. Bellovin, *Using Bloom Filters for Authenticated Yes/No Answers in the DNS*, Internet-Draft draft-bellovin-dnsext-bloomfilt-00, Internet Engineering Task Force (IETF), December 2001, Work in Progress.

[142] Dns-dir@ops.ietf.org Mailing List, *Using Bloom Filters with DNSSEC*, December 2001. Available at: https://psg.com/~randy/lists/dns-dir/msg00434.html [Accessed February 2024].

[143] Roland van Rijswijk-Deij, Gijs Rijnders, Matthijs Bomhoff, and Luca Allodi, *Privacy-Conscious Threat Intelligence Using DNSBLOOM*, in the Proceedings of the 16th IFIP/IEEE Symposium on Integrated Network and Service Management (IFIP/IEEE IM 2019), Washington, DC, USA, April 2019, pp. 98–106.

[144] Hachem Guerid, Karel Mittig, and Ahmed Serhrouchni, *Privacy-Preserving Domain-Flux Botnet Detection in a Large Scale Network*, in the Proceedings of the 5th International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, January 2013, pp. 1–9.

[145] PowerDNS Documentation, *Newly Observed Domain Tracking*. Available at: https://docs.powerdns.com/recursor/nod_udr.html [Accessed February 2024].

[146] *Random Subdomain Attack Mitigation Using Bloom Filter for Unbound*, GitHub Repository of Daisuke Higashi. Available at: https://github.com/hdais/unbound-bloomfilter [Accessed February 2024].

[147] Paul A. Vixie, Susan Thomson, Yakov Rekhter, and Jim Bound, *Dynamic Updates in the Domain Name System (DNS UPDATE)*, RFC 2136, April 1997. Available at: https://datatracker.ietf.org/doc/html/rfc2136 [Accessed February 2024].

[148] *Cuckoo Filter Implementation*, GitHub Repository of Huy Do. Available at: https://github.com/huydhn/cuckoo-filter [Accessed February 2024].

[149] *MurmurHash*, Wikipedia Article. Available at: https://en.wikipedia.org/wiki/MurmurHash [Accessed February 2024].

[150] *TXT Record*, Dynu Systems. Available at: https://www.dynu.com/Resources/DNS-Records/TXT-Record [Accessed February 2024].

[151] Harald Lang, Thomas Neumann, Alfons Kemper, and Peter Boncz, *Performance-Optimal Filtering: Bloom Overtakes Cuckoo at High Throughput*, in the Proceedings of the VLDB Endowment, Volume 12, Issue 5, pp. 502–515, January 2019.

[152] *Cuckoo Filters for Non Powers-of-2*, Computer Science Stack Exchange Forum Discussion. Available at: https://cs.stackexchange.com/questions/81876/cuckoo-filters-for-non-powers-of-2 [Accessed February 2024].

[153] *Internationalized Domain Names*, ICANN. Available at: https://www.icann.org/resources/pages/idn-2012-02-25-en [Accessed February 2024].

[154] Nguyen Van Tu, Jae-Hyoung Yoo, and James Won-Ki Hong, *Building Hybrid Virtual Network Functions with eXpress Data Path*, in the Proceedings of the 15th International Conference on Network and Service Management (CNSM 2019), Halifax, Canada, October 2019, pp. 1–9.

[155] Sebastiano Miano, Fulvio Risso, Mauricio Vásquez Bernal, Matteo Bertrone, and Yunsong Lu, *A Framework for eBPF-Based Network Functions in an Era of Microservices*, in IEEE Transactions on Network and Service Management (TNSM), Volume 18, Issue 1, pp. 133–151, March 2021.

[156] *Mizar Project*. Available at: https://mizar.readthedocs.io/en/latest/ [Accessed February 2024].

[157] Maximilian Bachl, Joachim Fabini, and Tanja Zseby, *A Flow-based IDS using Machine Learning in eBPF*, in arXiv Preprint arXiv:2102.09980, February 2021. Available at: https://arxiv.org/pdf/2102.09980.pdf [Accessed February 2024].

[158] *Whitepaper: DNS Reflection, Amplification, and DNS Water-Torture*, Akamai. Available at: https://www.akamai.com/content/dam/site/en/documents/research-paper/dns-reflection-vs-dns-mirai-technical-publication.pdf [Accessed February 2024].

[159] Keita Hasegawa, Daishi Kondo, and Hideki Tode, *FQDN-Based Whitelist Filter on a DNS Cache Server Against the DNS Water Torture Attack*, in the Proceedings of the 17th IFIP/IEEE International Symposium on Integrated Network Management (IFIP/IEEE IM 2021), Virtual Event, May 2021, pp. 628–632.

[160] *Chapter 6 Implementation*, GitHub Repository of Stavros Korentis. Available at: https://github.com/skorentis/dns_water_torture_xdp_mitigation [Accessed February 2024].

[161] *BPF Compiler Collection (BCC)*, GitHub Repository of the IO Visor Project. Available at: https://github.com/iovisor/bcc [Accessed February 2024].

[162] *Alexa Top 1 Million Sites*, Kaggle Dataset. Available at: https://www.kaggle.com/cheedcheed/top1m [Accessed February 2024].

[163] *DomCop Top 10 Million Domain Names*. Available at: https://www.domcop.com/top-10-million-domains [Accessed February 2024].

[164] *DGA Implementations*, GitHub Repository of Johannes Bader. Available at: https://github.com/baderj/domain_generation_algorithms [Accessed February 2024].

[165] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir, *NXNSAttack: Recursive DNS Inefficiencies and Vulnerabilities*, in the Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Virtual Event, August 2020, pp. 631–648.

[166] *SHAP: A Game Theoretic Approach to Explain the Output of Any Machine Learning Model*, GitHub Repository of "shap". Available at: https://github.com/shap/shap [Accessed February 2024].

[167] Tianqi Chen and Carlos Guestrin, *XGBoost: A Scalable Tree Boosting System*, in the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 2016, pp. 785–794.

[168] *Chapter 7 Implementation*, GitHub Repository of Nikos Kostopoulos. Available at: https://github.com/nkostopoulos/dga-explainability [Accessed February 2024].

[169] Xiaoqing Sun, Mingkai Tong, Jiahai Yang, Xinran Liu, and Heng Liu, *HinDom: A Robust Malicious Domain Detection System based on Heterogeneous Information Network with Transductive Classification*, in the Proceedings of the 22nd USENIX International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019), Beijing, China, September 2019, pp. 399–412.

[170] Karel Bartos, Michal Sofka, and Vojtech Franc, *Optimized Invariant Representation of Network Traffic for Detecting Unseen Malware Variants*, in the Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, August 2016, pp. 807–822.

[171] Franziska Becker, Arthur Drichel, Christoph Müller, and Thomas Ertl, *Interpretable Visualizations of Deep Neural Networks for Domain Generation Algorithm Detection*, in the Proceedings of the 2020 IEEE Symposium on Visualization for Cyber Security (IEEE VizSec 2020), Virtual Event, October 2020, pp. 25–29.

[172] Arthur Drichel, Nils Faerber, and Ulrike Meyer, *First Step Towards EXPLAINable DGA Multiclass Classification*, in the Proceedings of the 16th ACM International Conference on Availability, Reliability and Security (ACM ARES 2021), Virtual Event, August 2021, pp. 1–13.

[173] Nida Aslam, Irfan Ullah Khan, Samiha Mirza, Alanoud AlOwayed, Fatima M. Anis, Reef M. Aljuaid, and Reham Baageel, *Interpretable Machine Learning Models for Malicious Domains Detection Using Explainable Artificial Intelligence (XAI)*, in MDPI Sustainability, Volume 14, Issue 12, p. 7375, June 2022.

[174] Hatma Suryotrisongko, Yasuo Musashi, Akio Tsuneda, and Kenichi Sugitani, *Robust Botnet DGA Detection: Blending XAI and OSINT for Cyber Threat Intelligence Sharing*, in IEEE Access, Volume 10, pp. 34613–34624, April 2022.

[175] Giorgio Piras, Maura Pintor, Luca Demetrio, and Battista Biggio, *Explaining Machine Learning DGA Detectors from DNS Traffic Data*, in arXiv Preprint arXiv:2208.05285, August 2022. Available at: https://arxiv.org/pdf/2208.05285.pdf [Accessed February 2024].

[176] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin, *Why Should I Trust You?: Explaining the Predictions of Any Classifier*, in the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 2016, pp. 1135–1144.

[177] Sandra Wachter, Brent Mittelstadt, and Chris Russell, *Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR*, in Harvard Journal of Law and Technology, Volume 31, Issue 2, p. 841, Spring 2018.

[178] Omran Ayoub, Nicola Di Cicco, Fatima Ezzeddine, Federica Bruschetta, Roberto Rubino, Massimo Nardecchia, Michele Milano, Francesco Musumeci, Claudio Passera, and Massimo Tornatore, *Explainable Artificial Intelligence in Communication Networks: A Use Case for Failure Identification in Microwave Networks*, in Elsevier Computer Networks, Volume 219, p. 109466, November 2022.

[179] Daniele Scapin, Giulia Cisotto, Elvina Gindullina, and Leonardo Badia, *Shapley Value as an Aid to Biomedical Machine Learning: A Heart Disease Dataset Analysis*, in the

Proceedings of the 22nd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (IEEE/ACM CCGrid 2022), Taormina, Italy, May 2022, pp. 933–939.

[180] *Mozilla Public Suffix List*, Mozilla Foundation. Available at: https://publicsuffix.org/ [Accessed February 2024].

[181] Dingkui Yan, Huilin Zhang, Yipeng Wang, Tianning Zang, Xiaolin Xu, and Yuwei Zeng, *Pontus: A Linguistics-Based DGA Detection System*, in the Proceedings of the 2019 IEEE Global Communications Conference (IEEE GLOBECOM 2019), Waikoloa, HI, USA, December 2019, pp. 1–6.

[182] Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster, *Building a Dynamic Reputation System for DNS*, in the Proceedings of the 19th USENIX Security Symposium (USENIX Security 10), Washington, DC, USA, August 2010, pp. 273–290.

[183] Hong Zhao, Zhaobin Chang, Guangbin Bao, and Xiangyan Zeng, *Malicious Domain Names Detection Algorithm Based on N-Gram*, in Hindawi Journal of Computer Networks and Communications, Volume 2019, February 2019.

[184] *Wordninja*, GitHub Repository of Derek Anderson. Available at: https://github.com/keredson/wordninja [Accessed February 2024].

[185] Joewie J. Koh and Barton Rhodes, *Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings*, in 2018 IEEE International Conference on Big Data (IEEE Big Data 2018), Seattle, WA, USA, December 2018, pp. 2966–2971.

[186] Constantinos Patsakis and Fran Casino, *Exploiting Statistical and Structural Features for the Detection of Domain Generation Algorithms*, in Elsevier Journal of Information Security and Applications, Volume 58, p. 102725, January 2021.

[187] Wilhelm Kirch, *Pearson's Correlation Coefficient*, in Springer Encyclopedia of Public Health, 2008.

[188] Tahmina Zebin, Shahadate Rezvy, and Yuan Luo, *An Explainable AI-based Intrusion Detection System for DNS over HTTPS (DoH) Attacks*, in IEEE Transactions on Information Forensics and Security, Volume 17, pp. 2339–2349, June 2022.

[189] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer, *SMOTE: Synthetic Minority Over-sampling Technique*, in Journal of Artificial Intelligence Research, Volume 16, pp. 321–357, June 2002.

[190] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher,

Matthieu Perrot, and Édouard Duchesnay, *Scikit-learn: Machine Learning in Python*, in Journal of Machine Learning Research, Volume 12, pp. 2825-2830, November 2011.

[191] *XGBoost Library Documentation*. Available at: https://xgboost.readthedocs.io/en/stable/index.html [Accessed February 2024].

[192] *Keras - Deep Learning for Humans*, GitHub Repository of the Keras Team. Available at: https://github.com/keras-team/keras [Accessed February 2024].

[193] Petro Liashchynskyi and Pavlo Liashchynskyi, *Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS*, in arXiv Preprint arXiv:1912.06059, December 2019. Available at: https://arxiv.org/pdf/1912.06059.pdf [Accessed February 2024].

[194] Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee, *Consistent Individualized Feature Attribution for Tree Ensembles*, in arXiv Preprint arXiv:1802.03888, March 2019. Available at: https://arxiv.org/pdf/1802.03888.pdf [Accessed February 2024].

[195] Arthur Drichel, Justus von Brandt, and Ulrike Meyer, *Detecting Unknown DGAs without Context Information*, in the Proceedings of the 17th ACM International Conference on Availability, Reliability and Security (ACM ARES 2022), Vienna, Austria, August 2022, pp. 1–12.

[196] *Tranco Website: A Research-Oriented Top Sites Ranking Hardened Against Manipulation*. Available at: https://tranco-list.eu/ [Accessed February 2024].

[197] *GeForce GTX 1050 Ti Specifications*, NVIDIA Website. Available at: https://www.nvidia.com/en-gb/geforce/graphics-cards/geforce-gtx-1050-ti/specifications/ [Accessed February 2024].

[198] Wayne W. LaMorte, *Correlation Analysis*. Available at: https://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_correlation-regression/bs704_correlation-regression2.html [Accessed February 2024].

[199] *EarlyStopping*, Keras Documentation. Available at: https://keras.io/api/callbacks/early_stopping/ [Accessed February 2024].

[200] *BinaryCrossentropy Loss Function*, TensorFlow Documentation. Available at: https://www.tensorflow.org/api_docs/python/tf/keras/losses/BinaryCrossentropy [Accessed February 2024].

[201] *Adam Optimizer*, Keras Documentation. Available at: https://keras.io/api/optimizers/adam/ [Accessed February 2024].

[202] Daniel Plohmann, *DGArchive – A Deep Dive into Domain Generating Malware*, in Botconf 2015, Paris, France, December 2015. Available at: https://www.botconf.eu/botconf-presentation-or-article/dgarchive-a-deep-dive-into-domain-generating-malware/ [Accessed February 2024].

[203] *Matsnu Reverse Engineered Code*, GitHub Repository of Andrei Abakumov. Available at: https://github.com/andrewaeva/DGA/tree/master/dga_algorithms [Accessed February 2024].

[204] *eu.rvwgm2wrld2.xyz—How to Get Rid of It?*. Available at: https://easysolvemalware.com/eu-rvwgm2wrld2-xyz-how-to-get-rid-of-it/ [Accessed February 2024].

[205] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang, *Blockchain Challenges and Opportunities: A Survey*, in International Journal of Web and Grid Services, Volume 14, Issue 4, pp. 352–375, October 2018.

[206] Michael Nofer, Peter Gomber, Oliver Hinz, and Dirk Schiereck, *Blockchain*, in Springer Business and Information Systems Engineering, Volume 59, Issue 3, pp. 183–187, March 2017.

[207] Adam Pavlidis, Marinos Dimolianis, Kostas Giotis, Loukas Anagnostou, Nikos Kostopoulos, Theocharis Tsigkritis, Ilias Kotinas, Dimitrios Kalogeras, and Vasilis Maglaris, *Orchestrating DDoS Mitigation via Blockchain-based Network Provider Collaborations*, in The Knowledge Engineering Review, Volume 35, e16, pp. 1–17, April 2020.

[208] Alex D. Breslow and Nuwan S. Jayasena, *Morton Filters: Faster, Space-Efficient Cuckoo Filters via Biasing, Compression, and Decoupled Logical Sparsity*, in the Proceedings of the VLDB Endowment, Volume 11, Issue 9, pp. 1041–1055, May 2018.

[209] Thomas Mueller Graf and Daniel Lemire, *Xor Filters: Faster and Smaller Than Bloom and Cuckoo Filters*, in ACM Journal of Experimental Algorithmics (JEA), Volume 25, Issue 1, pp. 1–16, March 2020.

[210] Minmei Wang, Mingxun Zhou, Shouqian Shi, and Chen Qian, *Vacuum Filters: More Space-Efficient and Faster Replacement for Bloom and Cuckoo Filters*, in the Proceedings of the VLDB Endowment, Volume 13, Issue 2, pp. 197–210, October 2019.

[211] Giovane C. M. Moura, Sebastian Castro, John Heidemann, and Wes Hardaker, *TsuNAME: Exploiting Misconfiguration and Vulnerability to DDoS DNS*, in the Proceedings of the 21st ACM Internet Measurement Conference (ACM IMC '21), Virtual Event, November 2021, pp. 398–418.

[212] Petr Špaček, *NXNSAttack: Upgrade Resolvers to Stop New Kind of Random Subdomain Attack*, RIPE Labs, May 2020. Available at: https://labs.ripe.net/author/petr_spacek/nxnsattack-upgrade-resolvers-to-stop-new-kind-of-random-subdomain-attack/#:~:text=The%20main%20discovery%20in%20the,resolver%20as%20an%20amplifier%20for [Accessed February 2024].

[213] Sunil Kumar Singh and Pradeep Kumar Roy, *Detecting Malicious DNS over HTTPS Traffic Using Machine Learning*, in the Proceedings of the 2020 IEEE International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (IEEE 3ICT 2020), Virtual Event, December 2020, pp. 1–6.

[214] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon, *Federated Learning: Strategies for Improving Communication Efficiency*, in arXiv Preprint arXiv:1610.05492, October 2017. Available at: https://arxiv.org/pdf/1610.05492.pdf [Accessed February 2024].

[215] Arthur Drichel, Benedikt Holmes, Justus von Brandt, and Ulrike Meyer, *The More, The Better? A Study on Collaborative Machine Learning for DGA Detection*, in the Proceedings of the 3rd ACM Workshop on Cyber-Security Arms Race (ACM CYSARM '21), Virtual Event, November 2021, pp. 1–12.

[216] Marinos Dimolianis, Dimitris Kalogeras, Nikos Kostopoulos, and Vasilis Maglaris, *DDoS Attack Detection via Privacy-aware Federated Learning and Collaborative Mitigation in Multi-domain Cyber Infrastructures*, in the Proceedings of the 11th IEEE International Conference on Cloud Networking (IEEE CloudNet 2022), Paris, France, November 2022, pp. 118–125.

Index