BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER SYSTEMS
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

OPTIMIZATION OF DDOS MITIGATION RULE INFERENCE

OPTIMALIZACE ODVOZOVÁNÍ DDOS FILTRAČNÍCH PRAVIDEL

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ELENA CARASEC**
AUTOR PRÁCE

SUPERVISOR **MARTIN ŽÁDNÍK, Ing., Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Computer Systems (DCSY) Academic year 2021/2022

Bachelor's Thesis Specification

Student: **Carasec Elena**
Programme: Information Technology
Title: **Optimization of DDoS Mitigation Rule Inference**
Category: Networking

Assignment:

1. Nastudujte dostupnou literaturu o objemových DoS útocích a způsobech jejich potlačení s využitím strojového učení.
2. Seznamte se s nástroji a prostředím pro měření a analýzu síťového provozu v infrastruktuře CESNET.
3. Navrhněte možnosti optimalizace metody pro odvození pravidel pro blokování DoS paketů. V navržených optimalizacích uvažujte rozšíření metody, aby umožňovala nasazení do reálné infrastruktury.
4. Navržené optimalizace implementujte.
5. Implementaci ověřte v laboratorním i reálném prostředí, pokud to bude možné.
6. Zhodnoťte dosažené výsledky a diskutujte možnosti pokračování této práce.

Recommended literature:

- Dle pokynů vedoucího.

Requirements for the first semester:

- Splnění bodů 1 až 3 zadání.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Žádník Martin, Ing., Ph.D.**
Head of Department: Sekanina Lukáš, prof. Ing., Ph.D.
Beginning of work: November 1, 2021
Submission deadline: May 11, 2022
Approval date: October 29, 2021

Abstract

This thesis discusses the possibility of using machine learning algorithms for DDoS protection. For classical and incremental (online) learning are considered explainable supervised learning methods, particularly decision trees. Furthermore, some possible optimisations are introduced to increase traffic classification accuracy and decrease the amount of blocked legitimate traffic.

Abstrakt

Tato práce se zabývá možností využití algoritmů strojového učení pro ochranu proti DDoS útokům. Pro klasické a inkrementální (online) učení jsou uvažovány vysvětlitelné metody učení s učitelem, zejména rozhodovací stromy. Dále jsou představeny některé možné optimalizace pro zvýšení přesnosti klasifikace provozu a snížení množství blokovaného legitimního provozu.

Keywords

DDoS attack, filtration, machine learning, classification, supervised learning, incremental learning, data stream, decision tree, Explainable AI (XAI).

Klíčová slova

DDoS útok, filtrace, strojové učení, klasifikace, učení s učitelem, inkrementální učení, datový tok, rozhodovací strom, vysvětlitelná umělá inteligence.

Reference

CARASEC, Elena. *Optimization of DDoS Mitigation Rule Inference*. Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Martin Žádník, Ing., Ph.D.

Rozšířený abstrakt

Problematika kybernetické bezpečnosti dnes přitahuje pozornost každého, kdo používá výpočetní techniku, protože se nesmírně rozšířila a hraje zásadní roli v současném životě. Většina podniků, neziskových organizací a vládních služeb je do značné míry propojena s informačními technologiemi, což je činí efektivnějšími a více orientovanými na klienty. Jednu z největších hrozeb představují distribuované útoky typu odepření služby, které vyčerpávají výpočetní a paměťové zdroje oběti, až nebude schopna poskytovat potřebné služby. Největším problémem jsou relativně nízké náklady na DDoS útoky ve srovnání s výdaji na vytvoření ochrany před nimi.

Sdružení vysokých škol ČR a Akademie věd ČR (CESNET) vyvíjí projekt DDoS Protector, jehož cílem je zajistit adaptivní ochranu proti DDoS útokům. S důrazem na adaptaci je součástí DDoS Protectoru modul strojového učení, který na vstupu přijímá dva soubory ve formátu pcap (jeden s legitimními pakety a druhý s útočnými pakety) a jako výstup poskytuje odvozené filtrační pravidlo ve formátu BPF. Tato práce se zabývá návrhem a implementací modulu strojového učení a jeho optimalizací.

Současný prezentovaný návrh zahrnuje dvě možná řešení a obě se týkají rozhodovacích stromů, protože odkazují na vysvětlitelné strojové algoritmy, které jsou považovány za důvěryhodnější než jiné neprůhledné metody a lze je snadno interpretovat bez jakýchkoli matematických nástrojů. První přístup využívá klasickou offline metodu učení s učitelem z knihovny scikit-learn s názvem "Decision Tree Classifier". Další se učí inkrementálně na měnicích se datových tocích, který byl zaveden, aby se vypořádal s měnícími se vektory útoků pro přesnější předpovědi a odvodil pravidla relevantní v určitém okamžiku. Navržené optimalizace zahrnují ladění hyperparametrů pomocí mřížkového vyhledávání a zavedení parametru "Poměr legitimního provozu", který poskytuje modulu strojového učení dodatečné informace o procentuálním podílu legitimního a útočného provozu, aby se snížila míra legitimních paketů blokovaných odvozenými pravidly.

Experimenty provedené za účelem vyhodnocení výkonnosti navržených modulů ukazují, že algoritmus offline učení poskytuje lepší výsledky než metoda inkrementálního učení a je rychlejší, což je pro reakci na DDoS útoky zásadní kritérium. Nejlepší výsledky klasifikátoru rozhodovacího stromu jsou více než 98% skutečně pozitivních výsledků a méně než 3% falešně pozitivních výsledků. Zatímco "Hoeffding Adaptive Tree Classifier" vykazuje na stejných datových sadách po vyladění hyperparametrů 89% skutečně pozitivních a 6% falešných pozitivních výsledků.

Další testy probíhaly na různém poměru legitimního a útočného provozu po zahájení útoku. Algoritmus offline učení vykázal intuitivně srozumitelné výsledky, kdy se s poklesem míry útoku vysoce zvýšila míra falešně pozitivních výsledků, neboť se zvýšil i počet chybně označených legitimních vzorků. Zatímco inkrementální algoritmus vykazoval naprosto kolísavé výsledky, což znamená, že se na něj nelze spolehnout. Je patrné, že všechny legitimní vzorky jsou po zahájení útoku označeny jako útočné, protože jakékoli metody jejich oddělení a správného označení vedou k irelevantnosti aplikace metod strojového učení.

Optimalizace pomocí parametru míry legitimního provozu pomáhá snížit vedlejší účinek zmírnění útoku (blokování toků legitimních uživatelů), když útočné toky zabírají pouze 50% nebo méně příchozího provozu. Tato optimalizace snižuje blokování legitimního provozu z horní hodnoty 90% na 10%, což je znatelné zlepšení.

Přestože experimenty s klasifikátorem rozhodovacího stromu a jeho optimalizací prokázaly uspokojivý výkon, stále existuje prostor pro minimalizaci poměru legitimního provozu blokovaného odvozenými pravidly. Do té doby je třeba výstupní pravidla aplikovat obezřetně

a budoucí aktualizace algoritmu může být implementací úrovně důvěryhodnosti poskytované programem. Pravděpodobně by byla založena na míře podobnosti vzorků útočného provozu a poměru útočného a legitimního provozu v průběhu útoku.

Optimization of DDoS Mitigation Rule Inference

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Martin Žádník, Ing., Ph.D. The supplementary information was provided by Mr. Jan Kučera, Ing. I have listed all the literary sources, publications and other sources used during the preparation of this thesis.

.....
Elena Carasec
May 10, 2022

Acknowledgements

I would like to express my gratitude to my supervisor Martin Žádník, Ing., Ph.D., for his support, valuable advice, and regular consultations during all stages of this work. Also, I would like to thank the whole team working on the DDoS protector project for their willingness to help and provide all the necessary information.

Contents

Chapter 1

Introduction

In the last decades, informational and computational technologies have become vital in our lives. They bring a wide variety of possible applications such as online shopping, search engines, social media and cloud computing. IT helps businesses, governmental structures, and non-profit organisations be more efficient and client-oriented and benefits decision-making, inventory management and other data storage. It became accessible and widespread, and the significance of cybersecurity increased simultaneously.

One of the oldest and most dangerous types of cyberattacks is distributed denial of service (DDoS) attacks, which aim to cause a temporary or complete outage of the victim's systems. The risks for the victims are high because they are not able to provide services during an attack, which conduces to reputation loss and loss of trust from unsatisfied clients, and therefore financial problems. Arrangement of protection from DDoS attacks is costly, but DDoS-as-a-service can be purchased for a relatively small price.

This work aims to design and optimise the solution for automatised DDoS filtration rules inference using such machine learning algorithms as decision trees. The final program will become an ML module for the CESNET's DDoS Protector project. The module's goal is to infer blocking rules for attack traffic in BPF format, covering as less legitimate traffic as possible. This work compares two distinct approaches: one operates with classical offline supervised learning, and the other is online (incremental) learning on evolving data streams with drift detection.

Chapter 2 discusses the problematics of DDoS attacks, their types and existing prevention techniques and applications. Also, it describes the DDoS Protector project objectives and its inner structure. The following Chapter 3 presents the possibilities of machine learning methods use for automating filtration rules inference and discusses their benefits and shortcomings. Chapter 4 introduces the concept of the designed approaches for offline and online learning strategies using Decision Tree Classifier from scikit-learn and Hoeffding Adaptive Tree Classifier from River Python library correspondingly. Chapter 5 proposes certain optimisations for the methods from Chapter 4. Chapter 6 demonstrates the results of the designed programs and examines their performance. Finally, Chapter 7 deliberates the conclusions based on the information from 6 and suggests possible improvements.

Chapter 2

Distributed Denial of Service Attacks

This chapter discusses distributed Denial of Service attacks, why they represent a threat and the existing solutions to protect from them. Section 2.1 classifies the types of communication of a botnet. Section 2.2 analyses DDoS attack subtypes and specific attacks. Section 2.3 examines the current state of research on existing methods of protection from DDoS attacks, including prevention, detection and mitigation methods. The solution provided by CESNET is presented in Section 2.4.

2.1 Botnets and Communication

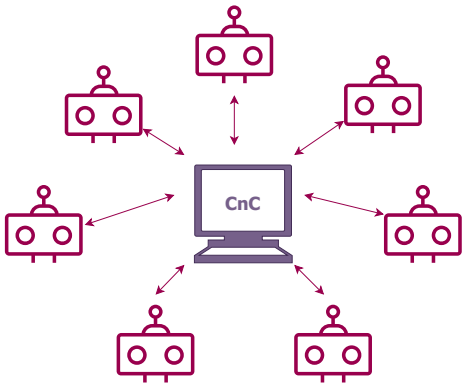
Denial of Service attacks and their distributed types constitute a category of cyberattacks, which aim to restrict access to a network resource or a machine for its use or even make it inaccessible to end-users. To prevent access, attackers flood the target system by sending an overwhelming number of requests, causing delays in responding to legitimate users or, ultimately, stopping responding. Another group of Denial of Service attackers tends to find vulnerabilities and exploit them to crash the target system. However, it is beyond this thesis's goals.

Distributed Denial of Service attacks are described as attacks in which multiple malicious systems are involved. They cooperate on performing a synchronised DoS attack that floods the bandwidth or other resources, including CPU time, RAM, or disk space. Assuming the distributed nature of the attacks, they are considered more dangerous than their non-distributed version because the number of generated traffic scales up with adding a new machine to the malicious network. Another problem with blocking such types of attacks lies in difficulties distinguishing malicious traffic from legitimate when coming from distributed resources that are likely to be spread throughout the world.

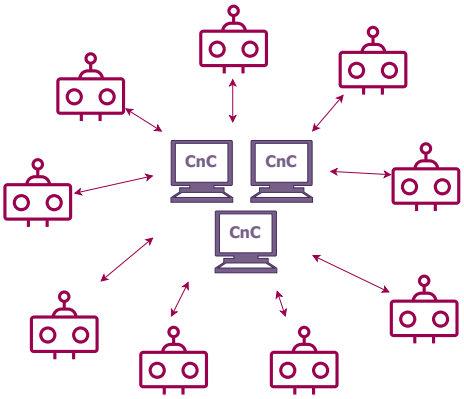
The basis of DDoS attacks are botnets. A botnet is a network of computers or other devices that have been infected with malware and remotely controlled by the main attacker, also called a „bot herder“. The term „botnet“ is a combination of the words „robot“ and „network“ [9], and the infected device is called a „bot“ or sometimes a „zombie“. To perform an attack, a botnet should have the ability to receive instructions from its bot herder, such as changing the target IP address, stopping an attack, or changing the attack pattern.

There are two primary communication design classes in a botnet: client/server and peer-to-peer models. The client/server botnet model resembles a remote workstation work-

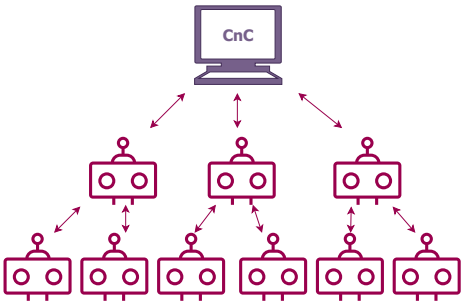
flow when each computer connects to a centralised server or one of the servers and obtains information from it. When all bots receive instructions from a command-and-control centre (CnC) resource like a web domain or an IRC channel, it is easy to update them since they should be changed only in the botnet's repository. The topology of the client/server botnet model can vary and includes the star, multi-server, and hierarchical network topology (Figure 2.1). Nevertheless, this approach has its shortcomings. The price of an uncomplicated information update is the vulnerability of the client/server model. For an attack to be blocked, only the server (or a group of servers in case of multi-server network topology) must be disrupted as the bots themselves cannot perform a severe attack without being controlled.



(a) Star network topology.



(b) Multi-server network topology



(c) Hierarchical network topology.

Figure 2.1: Client/server botnet model.

With the progress in attacks mitigation, botnets also evolve towards decentralisation. The peer-to-peer model lacks the imperfection of a centralised approach, so stopping these attacks is an arduous task. Each bot becomes both a command-and-control centre and a client simultaneously, so it receives updated instructions from its peers and simultaneously propagates them, as shown in the Figure 2.2. If all malicious information comes from a server, it is necessary to safeguard only it, which is typically strongly protected. However, in a decentralised network, bots are prone to becoming controlled by someone else, so botnet creators make them encrypted to limit potential access [9].

Although every business may become a victim of a DDoS attack, some areas are more frequently exposed to them. In 2021, the main targets were banks and financial institutions, but other businesses and industries are commonly targeted, including:

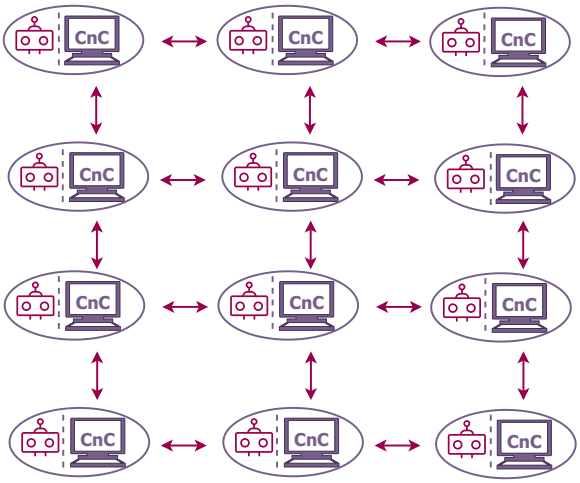


Figure 2.2: Peer-to-peer botnet model.

- educational institutions and remote learning services
- wired and wireless telecommunication carriers
- online gaming and gambling
- healthcare organisations
- governments and their agencies
- ISP, hosting, and related services
- technology companies [56].

2.2 Classification of Attacks

Undertaking an attack requires fewer resources than its prevention, especially with the proliferation of the Internet of Things (IoT) devices, which often come with default usernames and passwords that are not always changed [43]. Another reason why prevention of a DDoS attack is a challenge is a wide variety of DDoS attack subtypes. Even if a network is protected from a category of attacks by using mitigation strategies, it does not necessarily mean that it is protected from all of them.

Generally, DDoS attacks can be grouped into three categories:

1. Volumetric attacks
2. Protocol attacks
3. Application attacks

Volumetric attacks, also called volume-based, have the objective of saturating the victim's bandwidth by sending enormous volumes of traffic and creating traffic jams. The speed of volume-based attacks is measured in bits per second (bps). UDP, ICMP, and other spoofed packet floods represent this class of DDoS attacks.

Protocol Attacks are designed to consume the processing capacity of the network infrastructure, including servers, firewalls, and load balancers, sending malicious connection requests. Consequently, legitimate traffic cannot reach its target, and the victim may not even have the resources to respond to legitimate requests that have already reached it. Layer 3 and 4 requests attacks are measured in packets per second (pps). Protocol attacks are represented by TCP SYN, TCP ACK Attacks, Ping of Death, ICMP Attack, Smurf Attack, and more.

Application layer attacks, or Layer 7, mimic legitimate and innocent requests and exploit the weaknesses of Layer 7. They often open connections and initiate process and transaction requests, consuming finite resources such as the server's CPU time, disk space, and available RAM. Magnitude is measured in requests per second (rps). They are represented by low-and-slow attacks, GET/POST floods, attacks targeting Apache, Windows, or OpenBSD vulnerabilities [20][8].

Although this classification is the most common, alternative classifications can also be found. For instance, Hadeel S. Obaid classifies DDoS attacks into weakness-based and flooding attacks [42].

2.2.1 Specific DDoS Attack Types

Some of the most common and dangerous DDoS attacks are discussed below.

1. UDP flood attack. As its name indicates, during an attack, the attacker sends a massive number of User Datagram Protocol (UDP) packets to a specific or a random port to inundate it. Normally, when a UDP packet is received, the server tries to identify the application type on the specified port. In case when no application is associated with the port, it responds with ICMP Destination Unreachable message. The attacker continues to flood the victim using spoofed IP addresses until it is out of the available resources [37].

2. ICMP flood attack. This type is also called a ping flood attack. Generally, ICMP Echo Request packets are sent to check whether a remote host is alive. In case of success, it replies with ICMP Echo Reply, notifying about the possibility of establishing a connection. During a DDoS attack, ICMP Echo Request packets are sent with the broadcast destination IP address, so they are delivered to all the machines in the victim's network [37]. However, when the target is only one machine, the packet contains its specific address. Replying to these requests is very resource consuming.

3. TCP SYN attack. To establish a TCP connection before the data transmission, the client sends a SYN message to the server, which subsequently replies with a SYN-ACK message, and then the client acknowledges with the ACK message. It is called a three-way handshake. After starting an attack, the attacker sends many SYN packets again using spoofed IP addresses; the server fills in its table of TCP connections but never receives ACK messages, so all the connections are incomplete. Since the number of connections is limited, new legitimate users will not be able to establish connections with the victim server [62].

4. Ping of Death. An IP packet's maximum size is 65535 bytes. However, Data Link Layer (Layer 2) usually limits the maximum frame size; it can be, for example, 1500 bytes. Then a large IP packet is split into multiple packets called fragments, and the host reassembles them into a complete packet. However, in the case of a Ping of

Death attack, the initial packet is malformed somehow. The host gets an IP packet larger than 65535 bytes as a result, which leads to allocated memory buffers overflow and crashes the system [20].

5. DNS query flood. When a DNS resolver receives a request and has no response in its cache, it sends a request to a recursive DNS server. If the attacker plans his requests so that no addresses are known and kept in the server's cache, the victim will constantly send recursive requests. Taking into consideration the fact that recursive resolution is a pretty slow process, the servers quickly become overwhelmed [11].

2.3 DDoS Protection Techniques

DDoS defence mechanisms aim to protect victims from an immense amount of fake users packets. The following strategies ensure network protection from traffic inundation: prevention, detection, tolerance and mitigation, and response.

2.3.1 Prevention

Various prevention methods can be grouped into load balancing, honeypot, and sundry filtering mechanisms [36].

Load balancing is a resource distribution issue which pledges maximum utilisation of all the available resources. In general, the utilisation of network resources includes balancing between computing nodes named L7 balancing and balancing of network equipment, also called the L4 balancing technique [3].

Honeypots attract attackers, pretending to be a vulnerable part of the system to collect information about the initiated attack. They send responses imitating the existing system, making the attackers believe they succeed; meanwhile, they waste the resources. AmpPot [29] is an example of honeypots developed for amplification attacks monitoring. It allows observing the ongoing attacks and the techniques used by attackers. However, it stops responding when it is facing an attack itself. Ruchi Vishwakarma et al. [59] have presented a honeypot framework that uses machine learning techniques to derive information about attacks from the collected data logs. The advantage of collecting data from honeypots over datasets is that in case of a zero-day attack, the model will be learnt from the previously unknown attack types and find new patterns, based not only on the information of former well-known attack types.

Various filtering mechanisms are designed to help to prevent DDoS attacks. They include ingress filtering, cutting off traffic that does not match the domain prefix, egress filtering that forestalls the attack on other domains, and route-based and history-based packet filtering [36]. Cheng Jin et al. [22] have introduced Hop-Count filtering, which bases on the assumption that hop-count values are not consistent with the IP addresses at the moment of arrival to the victims. Abraham Yaar et al. [60] propose to filter traffic that was previously marked with a path identifier, which represents its path from source to a destination over the Internet. When a mark is identified as belonging to attack traffic, its part, defined by the threshold with the same mark, is dropped. As declared in the previous chapter, the attacks may be high-rate and slow-rate. Both need an individual approach for their recognition, so S. Toklu et al. have suggested a two-layer approach for filtering both high-rate and slow-rate attacks. High-rate DDoS attack flows are filtered using DAF (detection with average filter), and the rest of the traffic goes through DDFT (detection

with discrete Fourier transform) filters, which can block slow-rate attacks [57]. Tao Peng et al. [46] have introduced a History-based IP Filter mechanism for the edge-router: the router, which provides access to the Internet for the subnet, which is under defence. They propose to create an IP address database based on previous successful connections, which will become a kind of white-list in case of an attack. The advantage of this approach is the high confidence in filtration during DDoS attacks.

2.3.2 Detection

Dileep et al. [14] proposed the three classified categories of DDoS attack detection methods: statistical, knowledge-based, and soft computing.

Statistical methods compare new incoming instances to average traffic statistics and detect anomalies if the current traffic significantly differs. Feinstein et al. presented the solution based on entropy values, as the range shrinks when a network is under attack, and chi-squared statistics for discrete values [12]. Another entropy-based anomaly detection method involving a variation of Lyapunov exponent was introduced by Ma and Chen [35]. One of the most well-known statistically-based detectors is D-WARD [40], which is an inline system for an end network exit router. It collects per-destination and per-connection statistics for ingress and egress traffic.

Knowledge-based detection systems compare incoming traffic to already investigated attack patterns. Shabtai et al. [53] presented a knowledge-based temporal abstraction intrusion detection method on time-oriented data for mobile devices. Lin and Theng [31] propose to create detection knowledge from three sources: knowledge acquisition (KA) framework, domain experts and Characteristic Trainer. MULTIOPS [16] heuristics helps for attack detection by collecting data about traffic characteristics, where each network device maintains its data structure.

Dileep et al. [14] describe soft-computing methods as techniques that tolerate imprecision. Most of the approaches falling into this category are based on artificial intelligence algorithms. Pei et al. proposed a DDoS attack detection method based on random forest [45]. Meanwhile, Li et al. presented intrusion detection using neural networks [30].

2.3.3 Mitigation and Response

The practical strategies that are used as intrusion response are rate-limitation and filtration.

Malialis et al. [38] introduced a novel scalable reinforcement learning approach named Multiagent Router Throttling, which rate-limit traffic towards the victim server. Another distributed, coordinated, responsive method named ARROS [24] uses both of the techniques and can block or bandwidth-limit the intrusion. Kholidy et al. presented Autonomous Cloud Intrusion Response System (ACIRS) [26] providing defence for cloud systems. Fessi et al. [13] proposed a multi-attribute genetic algorithm model (MAGAM) for intrusion response. Yaar et al. [49] presented a Stateless Internet Flow Filter (SIFF), which classifies traffic flows into privileged and unprivileged. SIFF allows blocking individual flows selectively.

All three stages of DDoS defence are equally important. However, this work focuses on the last of them, aiming to present a filtration mitigation strategy for DDoS attack selective blocking using decision trees.

2.4 CESNET and „DDoS Protector“

Along with already presented commercial solutions for protection from DDoS attacks including AppTrana from Indusface, DDoS Attack Protection from Cloudflare, Azure DDoS Protection from Microsoft, DDoS-Guard, DDoS Protection from Akamai and many more, CESNET (Czech Education and Scientific NETwork) is developing its own product.

CESNET is an association of Czech universities and the Czech Academy of Sciences, which operates and develops the national electronic infrastructure for science, education and research, comprising a computer network and computational grids, data storage, and a cooperative environment.

Within the scope of the AdaptDDoS project [7] the DDoS protector was created with the support of the Ministry of Interior of the Czech Republic. This project aims to provide advanced protection from attacks on the availability of the services, which are widely known as DDoS attacks. CESNET's research and development team works on the project. It deals with the issues of adaptation to a changing vector of DDoS attacks, user procedure automation, and the use of information sources from external sources. The goal of the DDoS protector is not only to achieve a rapid and effective response to possible attacks from the perspective of accuracy but also to economise on financial expenses.

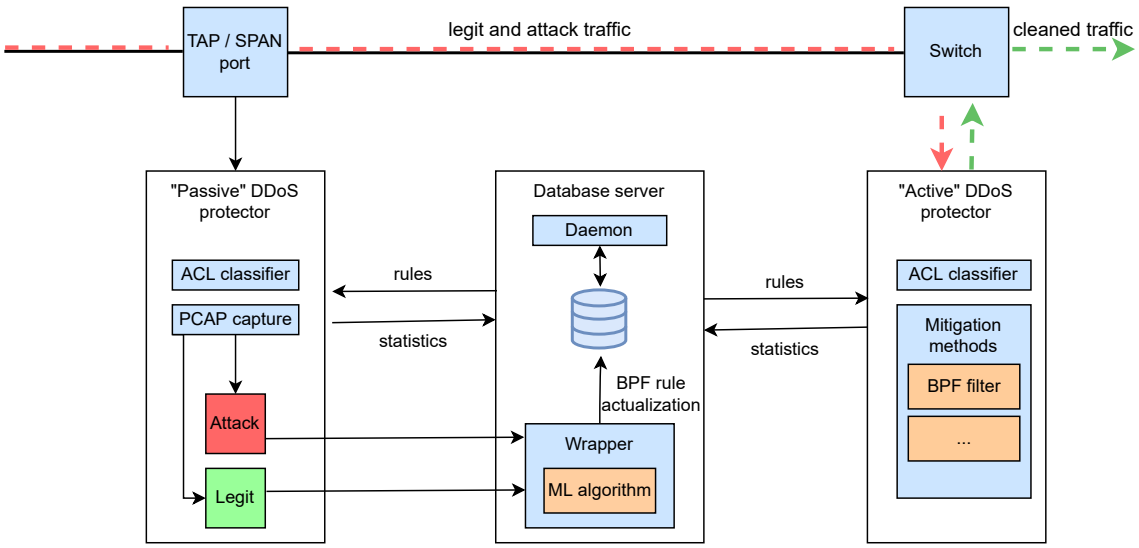


Figure 2.3: DDoS protector architecture.

The crucial feature of the DDoS protector is attacking traffic filtration, letting only legitimate traffic pass to the defended network. Figure 2.3 represents the inner architecture of the DDoS protector. It can work in two modes depending on the circumstances. If the traffic rate is lower than the network card can process (currently, CESNET's traffic filtering FPGA technology allows up to 400 Gb/s [25]), it drops no packets and only collects the statistics about the traffic for the future use. Whenever the traffic rate exceeds the threshold, declared the point when the defended network cannot process the current amount of traffic, the DDoS protector starts to clean traffic actively. Along with the other filtration methods, such as reputation IP address database from external sources, the machine learning algorithm starts to work taking as input two pcap (Packet CAPture) files: a legit file with previously collected packets and an attack file collected after the moment of exceeding the threshold limit. It is noteworthy that the attack pcap file will definitely contain legit

traffic since, after an attack starts, legitimate users still try to get the service. The output of the machine learning algorithm is a BPF rule derived from the tree structure of the model output, which is actualised every time a new attack pcap arrives. The BPF filtering rule declares which part of traffic should be dropped. Every time the database with rules is updated, the DDoS protector can work with the improved information to drop attack traffic with higher accuracy.

DDoS protector and all its modules are built using the DPDK (Data Plane Development Kit) framework[1], which is an open-source set of user-space libraries and drivers for network interface cards created by Intel and now managed by Linux Foundation. DPDK was designed to accelerate packet processing workloads running on various CPU architectures, including x86, ARM, and PowerPC.

[1] https://www.dpdk.org/about/news/

Chapter 3

Machine Learning for Traffic Filtration

Whenever the project aims to automate user processes, the use of artificial intelligence comes to mind. This chapter is about applying machine learning algorithms, a part of artificial intelligence, for DDoS defence. Section 3.1 covers the classification of machine learning algorithms. The role of explicability of machine learning algorithms is reviewed in Section 3.2 as long as the decision trees' key features. Section 3.3 explains algorithms of decision trees construction. The two approaches: offline learning and incremental learning, are discussed in Section 3.4 and from the perspective of their possible utilisation for attack traffic filtration.

3.1 Machine Learning Overview

Machine learning algorithms lie at the intersection of statistics and computer science. This area of study has become so prominent due to the following reasons: accessibility and a massive amount of available online data, and low computational costs [23].

As the classification of machine learning methods varies from one source to another, some categories are described in all, while others are less common. In this chapter, the ML methods classification overview will be in accordance with the book „Foundations of Machine Learning, Second Edition“ [41].

Machine learning algorithms differ in the types of available training data, the data receiving order and method, and the test data for evaluation purposes.

- **Supervised learning.** The learner obtains a set of labelled training data and makes predictions for previously unseen data. It is the most typical scenario for classification and regression problems. Spam detection is one of such problem example.
- **Unsupervised learning.** The learner receives only unlabelled data and tries to group them by features, which is the only available information. It is sometimes hard to evaluate the model as the test data are also unlabelled. The most common example of unsupervised learning is clustering.
- **Semi-supervised learning.** The learner obtains both labelled and unlabelled examples and tries to predict all the unseen points. These machine learning algorithms are popular in areas where it is easy to get unlabelled data and costly to label it. The expectations of this approach are based on the assumption that the distribution

of unlabelled data can help the learner achieve better performance than using less representative labelled data for supervised learning. Nevertheless, the conditions for obtaining better results using semi-supervised learning methods are in the research stage.

- **Transductive inference.** It is similar to semi-supervised learning because it also receives unlabelled and labelled samples. However, its goal is to label the unlabelled but previously seen examples. The circumstances under which the model will perform better are also under research, along with semi-supervised learning.

- **Online learning (incremental learning).** Contrary to the previously mentioned scenarios, the learner receives an unlabelled point, tries to predict it, and then learns its label. The aim is to minimise the cumulative loss, also called regret. The shortage of this approach is catastrophic forgetting because the model cannot keep information about all the examples as their number grows in time.

- **Reinforcement learning.** Similarly to online learning, training and testing rounds are intermixed in this type of learning. The learner actively interacts with the environment to collect the necessary information, gaining reward in passing for each action. It maximises the reward, taking actions and interacting with the environment. In reinforcement learning problems, the learner is faced with the exploration versus exploitation dilemma. There are two available strategies: collecting new unexplored information that may bring less reward if the current solution is optimal or exploiting the already collected information.

- **Active learning.** The learner interactively collects unlabelled training points, commonly by querying an oracle to label them. The aim is the same as in the case of semi-supervised learning: to obtain at least as good performance as in supervised learning but with fewer labelled examples. Since the learner chooses which examples to label, the number of points to be labelled by the oracle or teacher is much less than for supervised learning algorithms. For instance, active learning is used for computational biological applications.

3.2 Explainable Machine Learning

Nowadays, machine learning is widely used in different domains, including health care, entertainment and commercial, for various tasks, including recommendation systems, image annotation and classification, diagnosis prediction, and more [52]. Deep learning approaches perform better than humans in some types of tasks. At the same time, they have shortcomings, which are not only related to data quality, computational time and engineering efforts. The central problem is that these algorithms are incredibly opaque. Even though their mathematical background is understandable, deep learning approaches still suffer from a lack of declarative knowledge [18].

Because of social, ethical and legal reasons, it is desired for learning algorithms to provide comprehensive information about logical chains that influence their decisions, predictions, or actions [1]. According to European GDPR regulations, if any automated decision-making systems are used, the data subject should be able to receive meaningful information about the chain of reasoning behind that system and the possible consequences [6].

A.Adadi et al. [1] have formulated four main reasons for the need for explainable results of machine learning algorithms: to justify results, to control the system (to debug), for more straightforward model improvement, and gain new knowledge. Rationalising the results is helpful when the model makes unexpected decisions. Also, it ensures, if needed, that the outcomes are fair and ethical and that the algorithm complies with the criteria of Responsible Artificial Intelligence. Explainable machine learning enables enhanced control over the model's vulnerabilities and decreases the number of possible errors since they can be corrected fast. Moreover, it helps to improve the model when the information about the logic chain is not hidden by providing necessary additional data. New knowledge gain is impossible with the usage of only non-transparent algorithms. For example, it is desirable for a Go-player machine not only to win the game but also to describe its playing strategy. In the future, it would be highly beneficial if explainable models have the ability to derive new knowledge in natural sciences.

According to Belle V. et al. [2], the algorithms that are called transparent and, as a consequence, more trustworthy are:

- Logistic/Linear Regression
- Decision Trees
- K-Nearest Neighbours
- Rule-Based Learners
- Generative Additive Models
- Bayesian Models.

However, speaking about algorithmic transparency, most of the results of the algorithms from this list are too complex to be analysed without mathematical tools. The only class of algorithms a human without a mathematical background can understand is Decision Trees. Decision Trees are sequential models which contain a sequence of simple conditional statements; at each non-leaf node, a numeric attribute is tested against a threshold value or a set of possible values in the case of nominal attributes [28].

Considering the reasons mentioned above, including that the explainable and transparent models are the most trustworthy, they were chosen for DDoS defence goals. The results of Decision Trees can be converted to a set of traffic filtration rules, which are understandable to a network administrator without the need for complex mathematical tools and solid mathematical background.

3.3 Decision Trees

Inductive inference is the process of generalisation. The main objective is to learn how to classify objects using a given set of already labelled instances typically represented as attribute-value vectors. Each instance belongs to a class, and the task is to map unlabeled examples from attribute values to classes. Both labelled and unlabelled instances should be meticulously classified by this mapping [47].

A decision tree is a recursive structure expressing such mappings. It may consist of only one leaf associated with one class. However, commonly, such a tree consists of a sequence of test nodes, where each of them has a threshold, creating mutually exclusive outcomes,

and each of the outcomes also consists of similar sub-trees. For an object classification, it has to be tested by a path of testing nodes starting from the root until it reaches a leaf. Then the class label of the reached leaf will be assigned to the object [48].

An algorithm strives to do the best possible split for each iteration until it reaches stopping criteria, or no further split is possible because it does not lead to higher purity of the outcomes. If there is only one class in a leaf's subset, it is called pure; otherwise, it is impure. Each splitting iteration should lead to purer leaves of the decision tree as described in the Algorithm 25.

```
DT(Instances, Target_feature, Features)
    If all instances at the current node belong to the same category
    then create a leaf node of the corresponding class
    else
    {
        Find the features A that maximizes the goodness measure
        Make A the decision feature for the current node
        for each possible value v of A
        {
            add a new branch below node testing for A = v
            Instances_v := subset of Instances with A = v
            if Instances_v is empty
            then
            {
                add a leaf with a label the most common value of
                Target_feature in Instances;
            }
            else
            {
                below the new branch, add a subtree
                DT(Instances_v,Target_feature,Features - {A})
            }
        }
    }
```

Listing 3.1: A pseudo-code for building a decision tree [28].

The construction of decision trees is a challenging and computationally expensive problem. For a classification task regarded as tiny, the number of possible decision trees may be extremely high. For instance, in the case in which there are four discrete attributes, half of them with two possible values and the second half with three, and two classes, the number of decision trees exceeds $2.2 * 10^4$ [48].

Some types of heuristics were introduced to deal with complexity: heuristics based on information or entropy, heuristics based on error, and heuristics based on statistical significance [48]. ID3 and C4.5 algorithms use information gain for attribute selection, which is an entropy-based heuristic [19].

Information gain is calculated with the use of entropy, which has the formula:

$$Entropy = -\sum_{i=1}^{C} P(x_i) \log_2 P(x_i), \tag{3.1}$$

where C expresses the number of classes and $P(x_i)$ is the probability of randomly picking an element of class i.

$$IG(T, A) = Entropy(T) - \sum_{v \in A} \frac{|T_v|}{T} Entropy(T_v), \tag{3.2}$$

where T means target column, A is the feature (column), we are testing, and v means all possible values in A. If the result is positive, the split using the feature A lowers the entropy. The goal is to find the higher possible information gain value and do the split on that feature [33].

The CART algorithm uses **Gini index** for decision tree construction. The Gini index is calculated using the formula:

$$GiniIndex = 1 - \sum_{i=1}^{n} (P(x_i))^2, \tag{3.3}$$

where P_i is the probability of an element being classified for a specific class, same as in 3.1. The Gini index varies from 0 to 1 inclusive, where 0 stands for perfect classification. Consequently, a feature with less Gini index will be chosen for a split [58]. The Gini index belongs to the group of heuristics based on error.

Another splitting criterion, which is used primarily for imbalanced data streams, is **Hellinger distance**. Hellinger distance between two normal distributions P and N is defined as:

$$d_H(P, N) = \sqrt{1 - \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2}} e^{-\frac{1}{4}\frac{(\mu_1 - \mu_2)^2}{\sigma_1^2 - \sigma_2^2}}}, \tag{3.4}$$

where μ_1 is mean of P, σ_1^2 is its variance and σ_1 - standard deviation, and μ_2, σ_2^2 and σ_2 refer to N correspondingly [34]. It is used in online (incremental) learning by Hoeffding Trees also called Very Fast Decision Trees, which will be closely discussed in Section 3.4.

3.4 Incremental Learning and Online Decision Trees

Traditional machine learning methods work under the well-known scenario: all the data are in disposition before the training step. Therefore hyper-parameter setting and model selection takes into account the whole training dataset; meanwhile, the model in the training phase relies on the fact that the data is static [15]. However, in the context of network traffic and protection from DDoS attacks, the data is dynamic, and the attack vector may change anytime. The sector of machine learning algorithms which deals with evolving data streams is online (incremental) learning.

As opposed to traditional ML methods, an incremental model should be able to gather new knowledge from incoming data, memorise it and at the same time preserve the old one [32]. It should be able to make predictions based on the most recently available data at any moment. Furthermore, the most challenging is that it is lifelong learning on an infinite stream (without the ability to return to the already seen examples) that happens using limited resources such as time and memory [50].

Such decision tree algorithms as ID3 and C4.5 mentioned in Section 3.3 require all training data to be available before training starts. These methods cannot be applied to possibly infinite data streams, so they refer to offline learning algorithms. The opposites

to them are decision trees, which are capable of changing on the go with new incoming data, named online decision trees. The Very Fast Decision Tree (VFDT) is an illustrative example of online learners. It progressively grows with the new examples' arrival, and at the same time, it can reduce its branches' depth or turn its sub-tree into a node if necessary. The VFDT does not accumulate all the previously seen examples and considers only the actual ones. It helps to save memory resources and processing time [39]. However, it may cause problems if the behaviour of knowledge forgetting after concept drift is unsatisfactory.

A similar problem is described in the literature about deep learning algorithms such as neural networks. With the incoming of new data, the prior knowledge gets less weight by a backpropagation algorithm so that further information gets prioritised. It has an undesired impact on the performance of the previously learned examples. This effect is known as catastrophic forgetting. The model needs to be stable enough to sustain the current knowledge and sufficiently plastic to collect new information simultaneously for the effects of catastrophic forgetting reduction. Meeting both requirements at the same time is extremely challenging. This problem is called a stability-plasticity dilemma [32].

Neural networks severely suffer from the phenomenon of catastrophic forgetting. As a consequence, many articles about the possible ways of overcoming catastrophic inference (catastrophic forgetting) exist related to neural networks [17, 51, 27]. Parisi et al. [44] distinguish three ways of overcoming this phenomenon:

1. replay of previously learned knowledge, including rehearsal and pseudo-rehearsal,
2. regularisation approaches regarding the old knowledge weights,
3. network expansion.

In the context of incremental learning, replay techniques are also applicable. A.Robins [51] describes rehearsal as learning new information and model retraining on the previously seen data meantime. However, this approach has a constraint and is not suitable when old data is unavailable. Then another variant of replay called pseudo-rehearsal becomes useful. In a pseudo-rehearsal process, instead of actual examples, the model relearns using pseudo-items, which are artificially generated samples representing the classes (in classification problems). Pseudo-rehearsal approach becomes even more efficient with the increase in the number of target classes as rehearsal requires more memory space for keeping representative items of multiple classes.

Considering the fact that for DDoS protection purposes, only two classes named legitimate and attack will be identified, the rehearsal approach is applicable because of down-to-earth memory requirements.

Chapter 4

Design and Implementation of Filtration Rules Inference

This chapter describes the design of the algorithm for DDoS attack flow filtration. This work aims to propose and subsequently implement DDoS mitigation rules inference optimisations. This work is the continuation of the previous year's bachelor thesis named „Inference of DDoS Mitigation Rules“ [21]. Section 4.1 is dedicated to revising Jacko's design, which will also serve as an introduction to the problem. The author has implemented a program using the Decision Tree Classifier algorithm, which requires pcap files as input and generates Berkeley Packet Filter (BPF) filter rules for attack traffic as output. The features for the dataset are presented in Section 4.2. Section 4.4 describes the designed alternative to the classical supervised machine learning algorithms for DDoS protection presented in Section 4.3. It deals with the possibility of applying online learning methods to derail mitigating rules, particularly online decision trees. The output rules of both approaches are in BPF format.

4.1 Design Considerations

The primary source of information for this and the following section is [21] and the information provided by my supervisor.

The choice of using machine learning algorithms was influenced by the aim of reducing user intervention in the process of rules derivation. The intent of less user involvement in the mitigation process arose on the assumption of faster speed and more advanced generalisation abilities of machine learning algorithms compared to humans. When a DDoS attack starts, the speed of reaction to it plays a crucial role in the system's protection because a prolonged system outage discourages legitimate users from using the provided services again.

Machine learning methods are not always suitable. They may fail in non-deterministic problems, and they may not be applicable if a lack of data or, even worse, a lack of good data infers to receive statistically significant results [55]. The reasons why machine learning algorithms are considered to apply to the problem of DDoS mitigation rules inference are the following:

1. **Availability of data.** The input data for the inference is network traffic, where each packet represents a vector of information. The network flows can be monitored and stored in suitable file formats such as „pcap“ or „pcapng“. Consequently, the dataset is representative and is not prone to measurement errors.

2. **Peculiarities of attack traffic.** Assuming that the bots in a botnet, which send malicious packets to the victim, are headed by a central source of information, there may be distinctive features in the packets common either for all attack traffic or at least for its part.

3. **Distinctions between legitimate and attack traffic.** The features of legitimate traffic are supposed to be so diversified that it is not possible to find the common ones, or they will cover a huge part of attack traffic along with the legitimate. However, the expectation is that the attack traffic will differ from the legitimate in some features. Otherwise, the filtration rules will cover a considerable part of legitimate traffic, and it will lead to blocking access to benign users, which is the attackers' goal. That will mean that the attack is successful and the victim cannot afford it.

D.Jacko considers one more reason in his work [21]: **after an attack starts, the majority of packets in the traffic are actually sent by attackers**. Nevertheless, the experiments have shown that it is possible to detect attack flows using certain optimisations even if it is mixed with legitimate traffic in proportions, where legitimate traffic takes 50% and more. The designed optimisations are discussed with more details in Section 5.2.

The problem of mitigation rules inference can be solved using supervised learning algorithms due to the volumetric DDoS attack course. The first step is benign traffic capture. For the period when the traffic rate is lower than the defined threshold, the flows are labelled as legitimate. Even if there is a part of attack traffic, it cannot cause any damage to the system until it is capable of processing all the incoming packets at its usual speed. When the traffic rate exceeds the threshold, the system alerts an attack start. All the packets after that moment are labelled as attack. There might be a part of legitimate traffic during an attack, as declared above. However, due to the ML algorithm's generalisation abilities, it should be able to divide benign flows from the malicious if the amount of traffic captured in the peaceful period is massive enough and outweighs the number of legitimate packets labelled as attack. Immediately upon the fall of incoming traffic rate under the threshold, the traffic will be labelled as legitimate again, irrespective of whether the attack stopped or the inferred mitigation rules blocked it. This method of labelling packets allows receiving the two-class continuously updated dataset.

Decision trees were chosen for the DDoS protection among the supervised machine learning algorithms. The crucial role of explainable algorithms for trustworthiness in artificial decision-making agents was more thoroughly discussed in Section 3.2. In summary, people tend to be sceptical of the results of black-box algorithms as they might have a skewed view of the dataset and find only locally optimal solutions without generalisation attempts. Furthermore, even explainable machine learning algorithms' outputs may not be effortlessly understandable for people without specific qualifications. The majority of the algorithms' results should be interpreted using mathematical tools. On the contrary, the outputs of decision trees are understandable even for people without a mathematical background. They represent a sequence of decision nodes with the result at the leaf nodes. This characteristic is significant because the users of the designed program are supposed to be network administrators.

Furthermore, a considerable advantage of decision trees is that it does not require much data preparation. They need no data normalisation, one-hot encoding or blank values removal. Blank values are not expected to appear in examples constructed from packets, but lack of normalisation is fundamental for explicability. Due to all the reasons discussed

in this section, decision trees were chosen as the most suitable approach for DDoS rule inference.

4.2 Dataframe Structure

The input data for the designed program are pcap files with legit and attack traffic. They are parsed using Python module dpkt[1]. The information for the input dataset extracted from the pcap files concerns IP and transport layers (Layers 3 and 4). However, the application can be extended, and it is possible to extract information from other layers. The information is extracted from IPv4, TCP and UDP headers. The choice of protocols was primarily affected by the available datasets (pcap files) with attacking packets. The final dataframe contains data from the colored fields in Figures 4.1, 4.2 and 4.3.

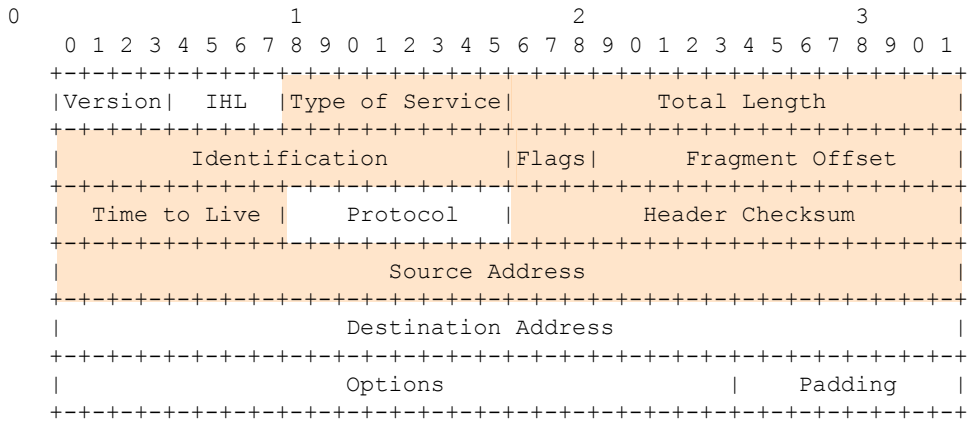


Figure 4.1: IPv4 protocol fields used to create the dataframe.

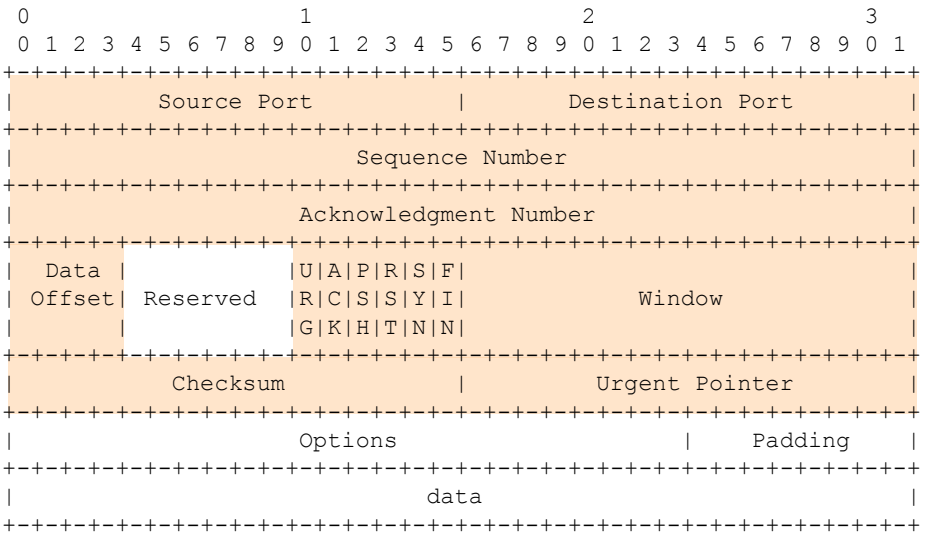


Figure 4.2: TCP protocol fields used to create the dataframe.

[1] https://dpkt.readthedocs.io/en/latest/

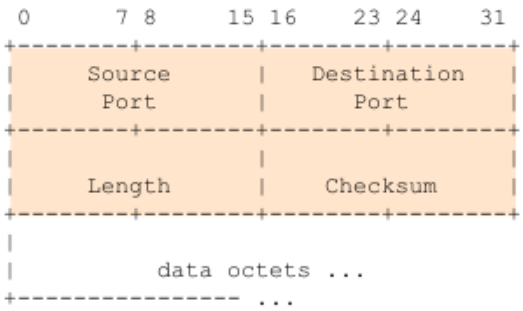


Figure 4.3: UDP protocol fields used to create the dataframe.

The destination IP address is not represented in the dataframe, even though this information can increase classification accuracy because the goal was to prove that the algorithm can find similarities in other fields. Otherwise, it would predominantly focus on the victim's address, making the task easier for the model.

The final dataframes always contain either columns from IPv4 with TCP or combine the columns from IPv4 with UDP. They are not joined into one dataframe because it would contain a lot of empty attributes; it would slow down processing time, which is unacceptable for mitigation rules inference. Moreover, with the addition of other protocol support, the number of blank values would increase proportionally.

4.3 Offline learning with Decision Tree Classifier

The next stage after the dataframe creation is model training. DecisionTreeClassifier[2] algorithm from scikit-learn [5] was chosen for the rules inference. Scikit-learn decision tree implementation uses an optimised version of the CART (Classification and Regression Trees) algorithm. The algorithm uses the feature and threshold for binary tree construction that brings in the maximal information gain at each node.

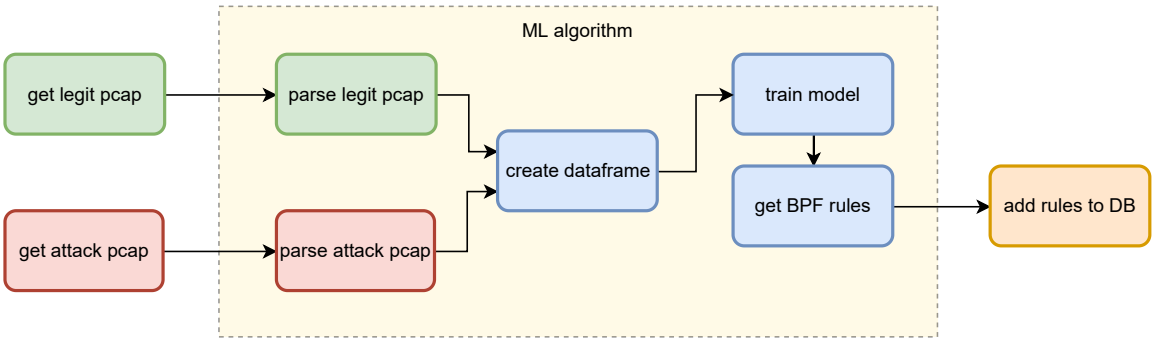


Figure 4.4: Scheme of rule inference using Decision Tree Classifier.

After the training step, the model is ready for making predictions and classifying examples. However, instead of proceeding to the testing phase, the final model is used to derail the rules, as Figure 4.4 shows. The tree structure is flattened and converted to a BPF filter. The testing stage is only used for the experiments, which are needed to prove the method's applicability and comparison of different approaches to tree construction.

[2] https://scikit-learn.org/stable/modules/tree.html#tree-classification

4.4 Online learning with Hoeffding Adaptive Tree.

Online or incremental learning is a contrasting approach to classical machine learning. It deals with sequential data, when the whole dataset either is not available at some point and the predictions should be made on an incomplete set, or when the flow of data is infinite, and a model cannot process it in a sensible timespan. Another reason for appealing to incremental learning methods is concept drift. Concept drift appears when the statistical properties of a target variable are unstable and unpredictably change over time. It is pertinent in the context of DDoS attack filtration as the vector of an attack may change anytime.

The Decision Tree Classifier cannot be used for data streams and requires the whole dataset to be available before the training step. Instead of it, we decided to use Hoeffding Adaptive Tree with Adaptive Windowing (HAT-ADWIN) [4]. Hoeffding Adaptive Tree is a method for decision tree construction, which has evolved from Hoeffding Window Tree (HWT). It can learn from a data stream without fixed window size specification. The perfect window size calculation is not a trivial task for users. It requires predicting the rate of distribution changes, and it gets more complex if the data stream changes are unforeseeable.

For the training step I used HoeffdingAdaptiveTreeClassifier[3] from River[4] library developed for streaming machine learning. It implements the Hoeffding Adaptive Tree algorithm with bootstrap sampling improvement, which helps to avoid overfitting.

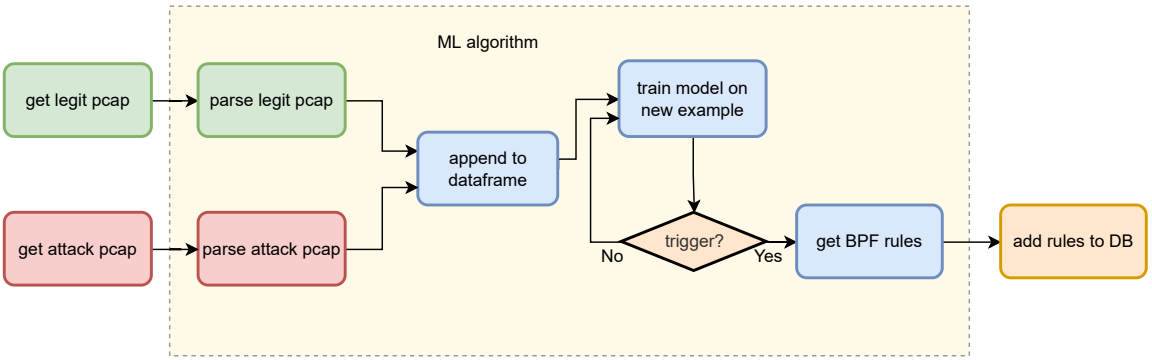


Figure 4.5: Scheme of rule inference using HAT-ADWIN.

The whole process of BPF rules inference using online is quite similar to the offline learning approach, which is visible from the comparison of Figures 4.4 and 4.5. Nevertheless, there are some slight differences. The first one is the opportunity of pre-training on the legit data before an actual attack starts, as the goal is to stop the attack as fast as possible. A further advantage is that the model does not store information about all the incoming packets but only about the most recent ones. It allows to save memory resources, and it helps to protect classification from wrong labelling in case of previous legit users' infection. In addition, the model will only consider actual attack flows if the vector of attack changes, which should increase the accuracy of the classification and decrease the number of output blocking rules. However, suppose the moment of ready rules in offline learning is determined as it is prepared right after training. In that case, the online model learns continuously, and it is not needed to have new rules after every learned example. It requires a kind of

[3] https://riverml.xyz/latest/api/tree/HoeffdingAdaptiveTreeClassifier/
[4] https://riverml.xyz/latest/

a trigger, which would enforce the model to pause learning and activate the conversion of the tree structure to BPF rules. This trigger may be in the form of a timeout, when the activation starts after a time period, for example, every 5 minutes. The alternative is to trigger when an event is happening, and it may be an attack start or attack vector shifting. This approach is more advanced and requires drift detection notifications, and was not implemented in this version of the program.

During the accuracy tests of HAT-ADWIN, the results of which will be discussed in Chapter 6, the problem of catastrophic forgetting appeared. After the model was trained on legitimate traffic examples, it learned attack examples. However, it unexpectedly started to completely forget the benign ones with the arrival of other attack samples and strengthening knowledge about them. The rehearsal approach is used to overcome the problem. At the stage of attack leaning, some legitimate random samples are injected in $5 : 1$ proportion. It reminds the algorithm that it should take into account the recently appeared attack examples and keep the information about the legitimate instances. Another possible solution is to mix all the available examples and learn the model on them, but it would be the same as using offline learning methods. It would undermine the whole point of using incremental learning algorithms and all the advantages it was meant to bring.

Chapter 5

Optimisations

This chapter will describe the suggested and implemented optimisations that improve both offline and online algorithms. They are used to increase the models' accuracy and decrease the rate of false positives. Section 5.1 introduces hyper-parameter tuning, and Section 5.2 proposes to provide additional information about the rate of legitimate traffic after the start of the attack to decrease the amount of legitimate traffic blocked by the inferred rules.

5.1 Hyper-parameter Tuning with Grid Search

In the „Inference of DDoS Mitigation Rules“ thesis are described various experiments with the hyper-parameters of Decision Tree Classifier are described, including „max_depth“, „min_samples_leaf“, „min_samples_split“ and „max_leaf_nodes“. The first proposed optimisation stands on the assumption that it would be beneficial to involve all the hyper-parameters, which may simultaneously influence the tree model's depth.

Scikit-learn library presents two different approaches as a solution to finding the best hyper-parameters combination: Grid Search and Randomized Search.

The grid search generates model candidates under the given lists of hyper-parameters values. After cross-validation for each model from the grid, the model with the best score is chosen for future testing. This approach may be computationally exhaustive since it fits each model from the grid on the training dataset.

Randomised search optimises the grid search algorithm, and each setting is sampled from a distribution over possible parameter values. Only a part of the model candidates will be investigated depending on the budget. This approach leads to a faster best model finding, skipping statistically unattractive ones. Among the advantages described by its authors is that additional parameters, which do not influence the performance, do not decrease the RandomizedSearchCV's efficiency.

Notwithstanding the above-presented advantages of the randomised search algorithm, GridSearchCV was chosen for optimisation. It allows to not only look up for the best possible model but also to investigate the trends in the output scores, which is meaningful, considering the possible different top hyper-parameters for different datasets from available attack pcap files. Another significant assumption for algorithm selection is the number of candidates in the grid. Since the grid for the experiments is four-dimensional, the time for the computations proliferates compared to the experiments with only one parameter at a time. Finally, it was stated that within the range of parameters from the previous

experiments, the calculation time for the grid search will be held during a reasonable time but will be more information-rich than the randomised search algorithm.

Various hyper-parameters for the HAT-ADWIN incremental learning model are also at the disposal to improve its performance. They differ from the Decision Tree Classifier except „max_depth“ and include „grace_period“, „split_confidence“ and „tie_threshold“. Unfortunately, an analogue of multi-core grid search is not implemented in the River library, and the available SuccessiveHalvingClassifier is not suitable for comparison to offline learning purposes. That is why the nested loops analogue of grid search was used for testing.

5.2 Legitimate Traffic Rate

5.2.1 The meaning of the „Legitimate rate“ parameter

When a tree classifier model is built, its output is converted into rules that declare which part of the traffic should be blocked. However, this set may cover more traffic than expected. It leads to a higher False Positive rate, which means that part of legitimate traffic will be blocked while trying to stop the attack, and the attacker has reached his goals.

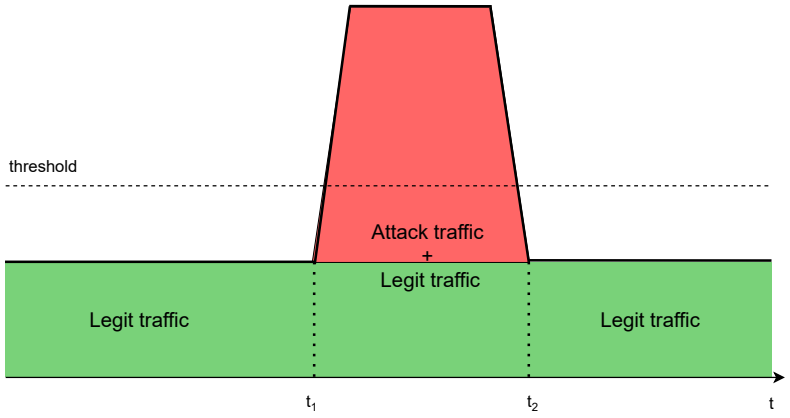


Figure 5.1: A simplified course of traffic during an attack.

A new parameter called the „Legitimate rate“ was introduced to decrease the number of legitimate packets marked as attack. This parameter declares the ratio of legitimate traffic to the overall traffic. Even though it is simply an input parameter, it would be helpful to know how it is calculated.

One of the crucial criteria of network systems is its bandwidth, which is typically measured in bits per second (bit/s or bps) or data packets per second (p/s or pps). Since the target for blocking are attack packet flows consisting of individual packets, it is appropriate to make measurements, particularly in data packets per second. As long as it is necessary to know the system's bandwidth, another measurement, the mean packet rate, will also come in handy for future calculations.

The mean packet rate value should be continuously calculated and monitored. So, when an atypical increase in the traffic rate is detected (as is shown as t_1 in Figure 5.1), it would be possible to determine the alleged ratio of legitimate traffic to the overall flow.

After the blocking rules are derived from the tree classifier, the next step is to compare the input „Legitimate rate“ parameter and the ratio of the legitimate leaves from the

resulting tree. If the second one is less than the input parameter, applying the rules will lead to a higher false positives rate because more traffic would be blocked than expected.

However, when the tree tends to block more packets than needed, it can be solved by picking a subset of rules that will „fit in“ the legitimate traffic rate and block the rest. Unfortunately, just sorting the rules by the number of legit packets in the resulting tree's leaves does not solve the problem because the leaves' impurity plays a crucial role here. The goal is to choose the purest nodes with the most legitimate packets and leave those least representing. This problem can be efficiently mapped to the „0/1 Knapsack problem“.

5.2.2 0/1 Knapsack problem

The 0/1 Knapsack problem is described as follows. A thief enters a store and sees n items. Each item has a value V_n, and its weight is W_n. He cannot take all of them because he can only carry a limited weight W in his knapsack. Of course, he wants to take as much valuable load as possible. Which items should he take and which leave? Note that the thief either takes an item or leaves it behind; he cannot take a fraction of an item nether take them twice or more times. The 0/1 points out this limitation [10]. Another version of this NP-full problem is the fractional knapsack problem, which is not suitable for the goals described in the subsection 5.2.1. It is impossible to pick a part of the rule because it would cause even a more extensive traffic part selection, or, in an edge case, it would not have any effect on blocking.

For the implementation of the knapsack problem, the dynamic programming algorithm[1] 16 was chosen, which finds the globally optimal solution in contrast to the Greedy approach.

```
def Knapsack(W, weights, values, n):
    K = [[0 for x in range(W + 1)] for x in range(n + 1)]

    for i in range(n + 1):
        for w in range(W + 1):
            if i == 0 or w == 0:
                K[i][w] = 0
            elif weights[i-1] <= w:
                K[i][w] = max(
                    values[i-1] + K[i-1][w-weights[i-1]],
                    K[i-1][w])
            else:
                K[i][w] = K[i-1][w]

    return K[n][W]
```

Listing 5.1: Dynamic programming algorithm for 0/1 Knapsack problem.

Regarding the knapsack problem, n is the number of rules. The „weight“ is the number of packets in the corresponding rule for each rule. As mentioned above, the purest nodes with the most examples should be picked, so the formula 5.1 was proposed, where x is impurity and y is the number of legitimate packets in the node.

[1] https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

$$
\begin{aligned}
V(x,y) &= x^2 y \\
&= \left(\frac{leaf\_legit\_examples}{leaf\_legit\_examples + leaf\_attack\_examples} \right)^2 leaf\_legit\_examples \\
&= \frac{leaf\_legit\_examples^3}{(leaf\_legit\_examples + leaf\_attack\_examples)^2}
\end{aligned}
\tag{5.1}
$$

This formula emphasises the importance of the impurity parameter; nevertheless, it involves the number of legitimate packets, too. Otherwise, the algorithm would first choose the purest rules with the least number of examples in the node.

Chapter 6

Experiments and Evaluation

This chapter describes the procedure of experiments and evaluation methods. They are necessary to prove the design's applicability or disprove the hypothesis about the possibility of benefiting from specific approaches for filtration rules inference. Section 6.1 concerns the data for test conduction. Evaluation metrics are introduced in Section 6.2, while the results of individual tests are discussed in Sections 6.3 and 6.4. A great emphasis is put on the false positives rate in Section 6.5 with the evaluation of the results. The main goal is not to let the attackers block access to legitimate users, and it will not be reached if the machine learning algorithm blocks the benign flows itself.

6.1 Datasets

All the experiments were held on the same datasets, which are described in [21]. The intention was to make the tests as closer to reality as possible. The legitimate traffic was captured on the core network between the Austrian national research and education network (ACONET) and Czech Education and Scientific Network (CESNET). The captured traffic pcap was split into two separate pcap files representing the course of an attack, as Figure 5.1 demonstrates. The first part, named LEGIT, contains about 76800 packets, where TCP:UDP ratio is about 4 : 6 and represents legitimate traffic until $t1$, and the second part, named MIX, represents the legitimate traffic flows between $t1$ and $t2$. All the LEGIT pcap samples are marked as „legitimate“ (encoded as „0“) for both the training and testing stages. Even though the MIX pcap contains legitimate packets, we treat them differently. After the start of an attack, surpassing the threshold, all packets are labelled as „attack“ (encoded as „1“). Both datasets, attack and MIX, are assumed as attack for training. However, for the testing stage, the packets from the MIX pcap are labelled as „legitimate“. It helps to assess the decision tree's generalisation abilities properly. All the attack samples are labelled as „attack“ for the training and testing stages.

A part of the attack datasets is from CICDDoS2019 dataset [54], which Canadian Institute for Cybersecurity published. It contains 13 different types of DDoS attacks. SYN flood, UDP flood, DNS and NTP amplification were picked. The second part was obtained by using tools for DDoS attack generation. These tools are LOIC[1], HULK Python script[2] and Torshammer[3]. As the datasets contained only a few source IP addresses, the addresses

[1] https://en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon
[2] https://allabouttesting.org/hulk-ddos-tool-complete-installation-usage-with-examples/
[3] https://sourceforge.net/projects/torshammer/

Table 6.1: Multi-vector attack datasets.

Name	Mix of datasets
SYNDNS	SYN,DNS
ALLUDP	DNS, LOIC, NTP, UDP
ALLTCP	TORSHAMMER, SYN, HULK
ALL	DNS, LOIC, NTP, UDP,TORSHAMMER, SYN, HULK

were changed to random to make the task more complicated. Each attack pcap file contains 4000 packets. Since the results of the tests on each particular attack type are too good, they were joined into multi-vector attack datasets according to the Table 6.1.

As Figure 6.1 depicts, for the training dataset creation, 60% of the LEGIT dataset and 60% of the attack dataset are used, and 40% of the LEGIT and attack datasets are used for testing correspondingly. The number of packets from the MIX dataset is chosen as $3:7$ to the attack packets for most experiments, so the attack traffic consumes 70% of the resources. All the packets from the MIX dataset created for training are used for testing.

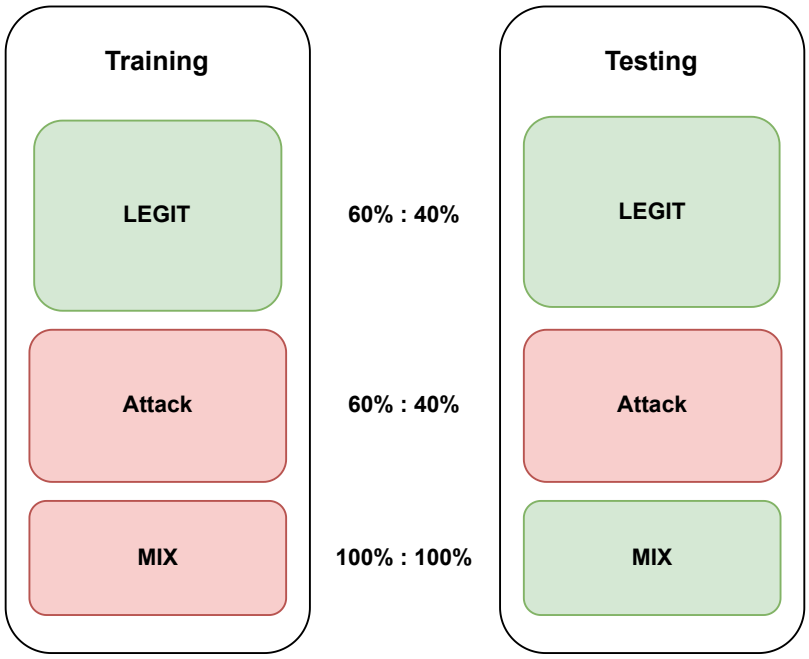


Figure 6.1: Dataset mix of legitimate and attack traffic.

6.2 Evaluation Metrics

To evaluate the classification performance of a model and draw a comparison between models with different parameters, all ML libraries implement a confusion matrix 6.2. It keeps the information about the number of wrongly and rightly classified samples. Absolute numbers from the matrix were converted to percentages, where each row adds up to 100%.

The graphs in the following sections depict True Positives and False Positives evolution using different parameters.

		Predicted	
		Positive (1)	**Negative (0)**
Actual	**Positive (1)**	True Positive (TP)	False Negative (FN)
	Negative (0)	False Positive (FP)	True Negative (TN)

Table 6.2: Confusion matrix.

For the experiments with grid search, the comparison based on two parameters simultaneously is impossible. Only one parameter named score is needed. There are many various scoring formulas in the scikit-learn library. However, they do not meet our requirements, as the aim is to have as low False Positive rate as possible and a reasonably high True Positive rate. The formula used for scoring is:

$$\begin{aligned} Score &= \frac{\frac{1}{3}(TP + TN + (100 - FN)) + 3(100 - FP)}{4} \\ &= \frac{\frac{1}{3}(TP + TN + TP) + \frac{9}{3}TN}{4} \\ &= \frac{2TP + TN + 9TN}{12} \\ &= \frac{TP + 5TN}{6} \end{aligned} \tag{6.1}$$

The formula 6.1 emphasises that right labelling of legitimate packets is 5 times more important than right labelling of attack packets.

6.3 Decision Tree Classifier

This section describes the results of experiments with the Decision Tree Classifier from the Python scikit-learn library.

Figures 6.2–6.5 depict True Positive and False Positive rate for the „max_depth“, „max_leaf_nodes“, „min_samples_leaf“ and „min_samples_split“ parameters. Each figure demonstrates the results of testing on 4 available datasets: ALL, ALLTCP, ALLUDP and SYNDNS from the Table 6.1. All the tests on individual parameters use the datasets where attack to MIX ratio is 70% : 30%.

The shown values of True Positives vary from 80% to 100% because the values under 80% are unsatisfactory. The shown values of the False Positive rate are up to 15% because this range covers the majority of the results, and at the same time, higher values are unacceptable.

6.3.1 Maximum Depth

The first examined parameter is the maximum depth of the tree. The choice was based on the assumption that a tree with unlimited depth is predisposed to overfitting and thus a high False Positives rate for testing because of the MIX dataset. The hypothesis has been confirmed. As Figure 6.2 demonstrates, the increase of allowed maximum depth also heightens the False Positive rate for all the available datasets. The same goes True Positive

rate, growing up to 100% in all cases. If the tree is too limited and has a maximum height of less than 5, it cannot take the uttermost from the available features.

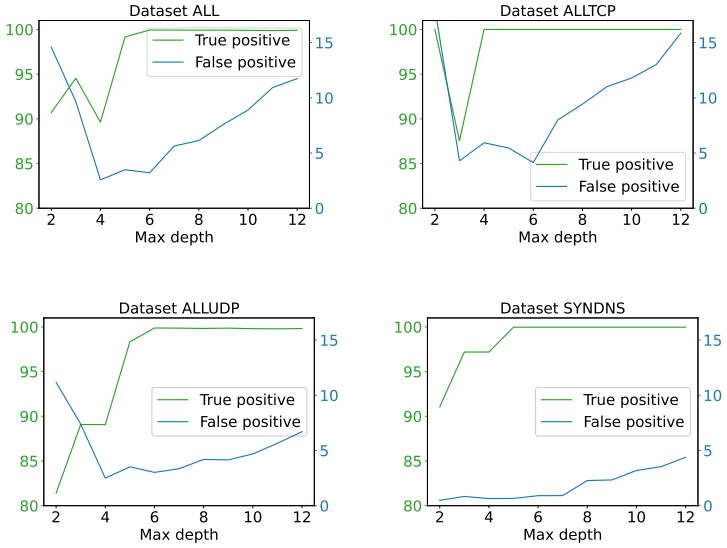


Figure 6.2: Decision Tree Classifier: maximum depth.

6.3.2 Maximum leaf nodes

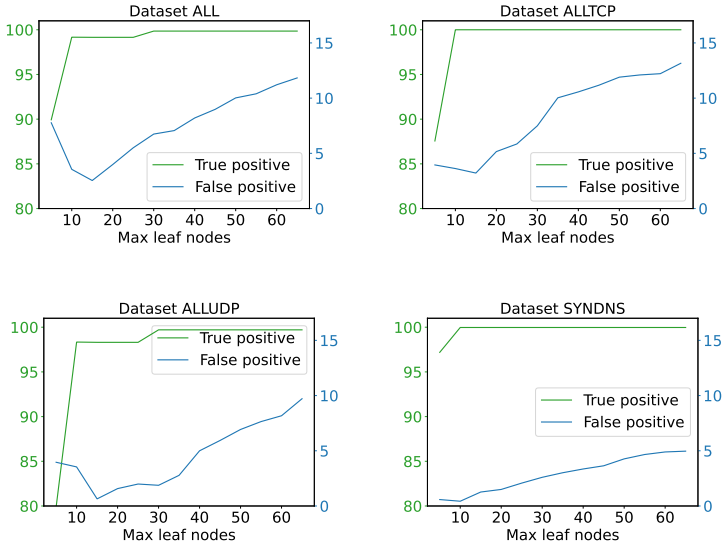


Figure 6.3: Decision Tree Classifier: maximum leaf nodes number.

A similar situation is with the „max_leaf_nodes“ parameter, which limits the number of leaf nodes instead of the depth. A higher parameter value also raises True Positive and False Positive rates, which is visible from Figure 6.3. The advantage is that the tree can construct a long sequence of decision nodes if needed according to the values of the features. However,

they are limited in the amount and are not likely to overfit. A tree with the maximum depth n can have up to 2^n leaf nodes. Nonetheless, a Decision Tree Classifier with a maximum depth of 4 shows relatively low performance, and a tree with 16 leaves is close to the best possible results for all datasets.

6.3.3 Minimum samples in a leaf.

These experiments concern the „min_samples_leaf“ parameter. The aim is to discourage the model from creating too pure leaves, which, again, due to the labelling of the MIX dataset, may lead to a high False Positives rate. The parameter's values presented in Figure 6.4 vary from 0% to 6,5% of the training dataset. The results are not so unequivocal as for the previous two tested parameters. Opposite to the supposition, in all the datasets except SYNDNS, the most uncomplicated attack to detect, the False Positive rate tends to increase as long as the True Positives rate decreases. The optimal value appears to be about 2%.

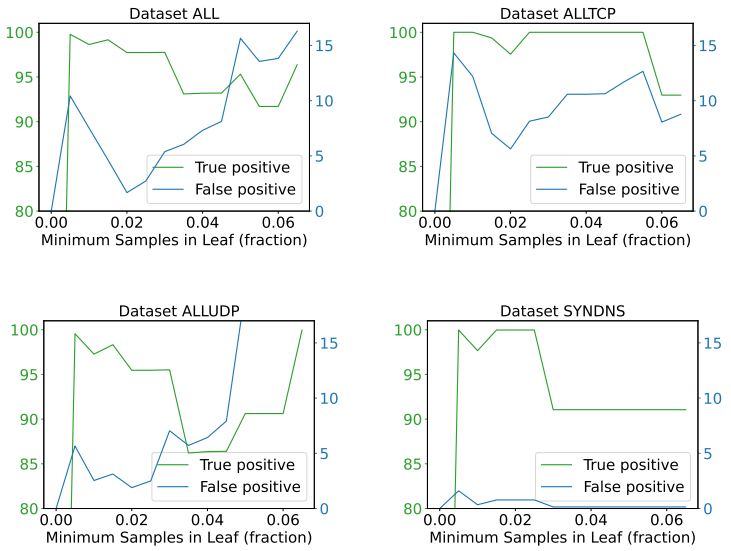


Figure 6.4: Decision Tree Classifier: minimum samples in a leaf.

6.3.4 Minimum samples for a split.

The last examined parameter of the Decision Tree Classifier is „min_samples_split“. The supposition about its adjustment was similar to the „min_samples_leaf“. A node cannot be split until it has enough samples to do it. If the number is too small, the model will overfit; if it is too big, the model will underfit. The tested percentages are again up to 6,5%. Here, the results were more predictable than for the last parameter. A more considerable fraction leads to less False Positive, keeping a high True Positive. However, a fraction bigger than 5% leads to a False Positives increase because the model cannot be specific enough to make precise predictions as Figure 6.5 depicts.

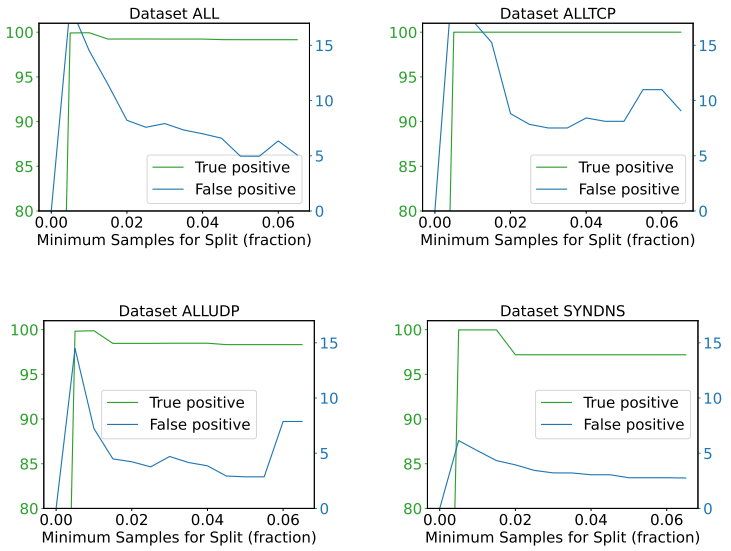


Figure 6.5: Decision Tree Classifier: minimum samples in a leaf for a split.

6.3.5 Grid Search

One of the designed optimisations was choosing the best parameters using grid search. It allows to combine of individual parameters' advantages and thus achieves a better performance score. As Table 6.3 demonstrates, the achieved results are above 98% of the True Positive rate and under 3% of the False Positive rate. The experiments have shown that maximum depth restriction is not that important, and other parameters may successfully limit the model. The maximum leaf nodes limitation plays a crucial role here and prevents overfitting. The best result of 10-15 nodes is similar to the best score from testing the parameter „max_leaf_nodes“ individually. The most unpredictable hyper-parameter minimum samples in a leaf turned out to be left closer to its default value of 1. At the same time, the results of the best minimum samples for a split parameter in combination with the other parameters do not coincide with testing the hyper-parameter individually. The overall tendency is to enforce generalisation for a multi-vector attack using „min_samples_split“ together with „max_leaf_nodes“. The evidence of the maximum leaf nodes' significance is valuable knowledge because the consequence is that the resulting number of BFP rules concatenated in one rule will always be lower than 15.

Dataset	**max_ depth**	**max_ leaf_ nodes**	**min_ samples_ leaf**	**min_ samples_ split**	**Results (TP, FP)**
SYNDNS	≥5	10	0.005	0.005	99.94%, 0.93%
ALLUDP	≥7	15	0.005	0.01	98.31%, 0.64%
ALLTCP	≥5	10	0.005	0.06	100%, 0.94%
ALL	≥7	12	0.005	0.005	98.32%, 2.86%

Table 6.3: Decision Tree Classifier: grid-search results for the individual datasets.

6.3.6 MIX to attack samples ratio

This section of experiments aim to examine Decision Tree Classifier's performance with different ratio of MIX dataset size to attack datasets. The bigger is the percentage of samples from the MIX dataset, more truly legitimate packets are marked as attack. For this bunch of experiments, the model with the following parameters was used:

- max_depth = 10
- max_leaf_nodes = 12
- min_samples_leaf = 0.005
- min_samples_split = 0.005

The choice of the values was influenced by grid search results, where models with these parameters resulted in the mean best performance. Following the supposition, the True Positive rate is relatively stable because the actual attack dataset does not change in these experiments. At the same time, the increase in the percentage of the MIX dataset also grows the False Positive rate because the LEGIT dataset cannot overweight it after a certain percentage, about 50%, anymore. The overall tendency is that with more samples in the LEGIT dataset and the simpler the attack vector is, the lower is False Positive rate, which is noticeable in Figure 6.6.

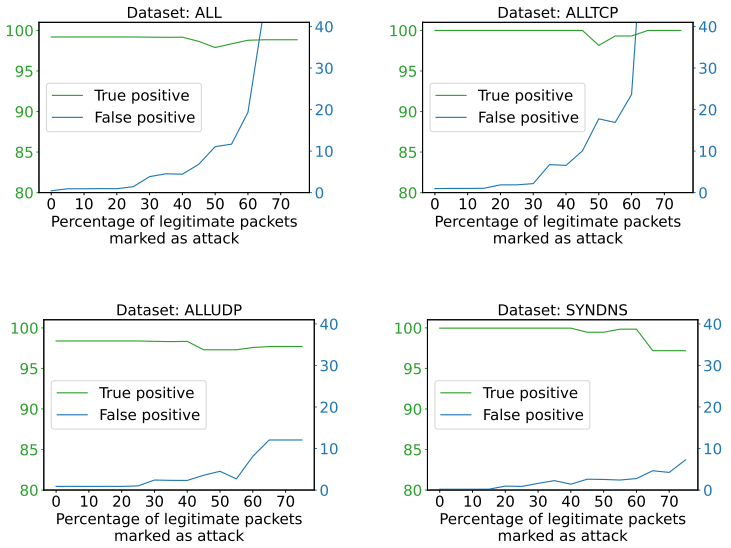


Figure 6.6: Decision Tree Classifier: percentage of legitimate packets marked as attack.

6.3.7 Knapsack results

The optimisation was implemented to reduce the False Positive rate, which provides additional information to the model about the actual rate of attack packets using the 0/1 knapsack algorithm. Comparing Figures 6.6 and 6.7, we see a considerable decrease in the False Positive rate for all datasets when the attack rate is 50% or less, keeping attack detection with more than 50% rate still on quite sensible values. We observe an abrupt

decrease in the True Positive rate at some points, which is a disadvantage of this approach. Consequently, it is better to use this optimisation only when the attack rate is less than 50%.

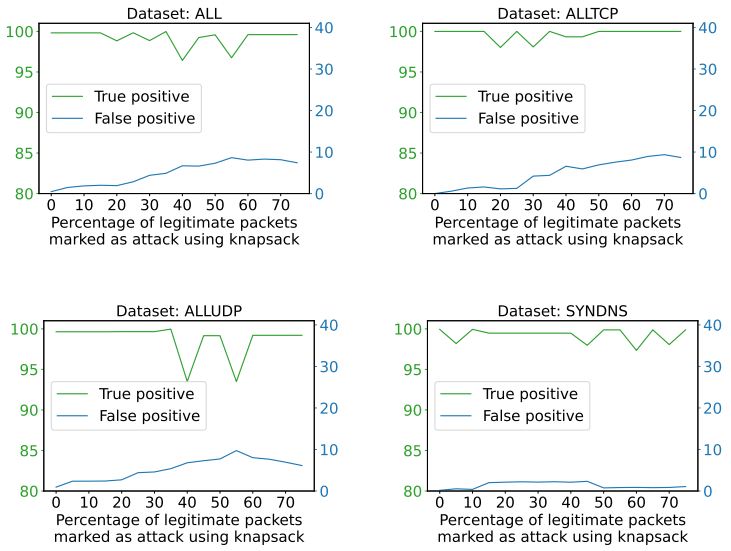


Figure 6.7: Decision Tree Classifier: percentage of legitimate packets marked as attack with 0/1 knapsack optimisation.

6.4 Hoeffding Adaptive Tree

The following group of experiments were held to understand if decision trees for evolving data streams are applicable for solving the problem of attack and benign packet classification. First of all, the results of using different values of positives weight parameter are discussed to overcome the catastrophic forgetting. Figures 6.9–6.12 demonstrate the results of experiments on individual parameters of the Hoeffding Adaptive Tree Classifier from the River library using the best found „positives_weight“ parameter value. These four hyper-parameters are presented because the experiments with the others showed flat graphs and did not influence the results. The last two experiments are the same as were described for the Decision Tree Classifier: test the model on different MIX to attack dataset ratio and test the „Legitimate rate“ using 0/1 knapsack algorithm optimisation.

6.4.1 Positives weight

The „positives_weight“ parameter is not one of the hyper-parameters of the Hoeffding Adaptive Tree Classifier. It means the weight of each sample labelled with „0“ or „legitimate“, and the negative weight is $1 - positive\_weight$, correspondingly. This parameter was introduced to balance the rate of legitimate samples during rehearsal as described in Section 4.4. As the number of LEGIT samples during the rehearsal is less than the number of attack samples, only values equal to or greater than 0,5 are presented.

Figure 6.8 demonstrates that the True Positive rate value is higher when the parameter's value is closer to 0,5. However, the False Positive rate results at that point are unacceptable.

The optimal value of the „positives_weight“ is 0,65 because with greater values come True Positives rate fall-down. This value is used for the subsequent experiments with Hoeffding Adaptive Tree Classifier's hyper-parameters.

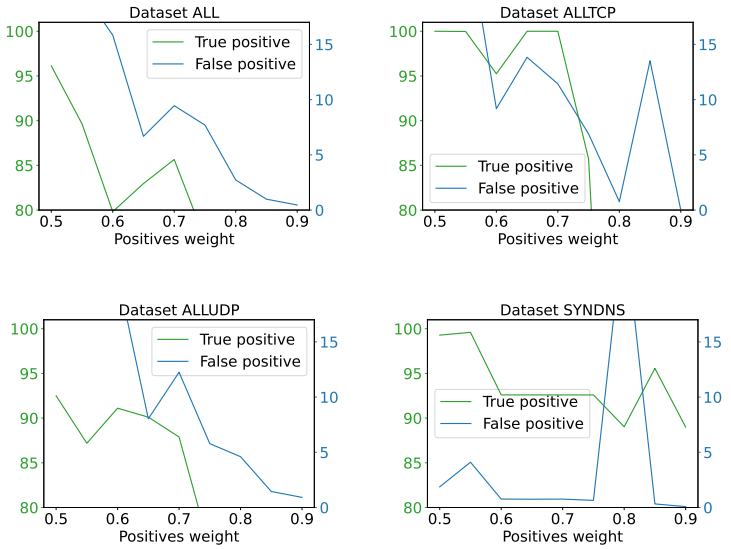


Figure 6.8: Hoeffding Adaptive Tree Classifier: positives weight.

6.4.2 Maximum depth

The maximum model's depth is the only identical hyper-parameter of the Decision Tree Classifier and Hoeffding Adaptive Tree Classifier. Again, Figure 6.9 shows that, when the tree is too restricted and has less than five nodes in a sequence allowed, it poorly recognises the attack flows and suffers from underfitting. It finds its optimum at six and does not require future limitations because of pre-pruning, presumably.

6.4.3 Grace period.

The „grace_period“ hyper-parameter value tells how many instances a leaf observes before doing a split. It is pretty similar to the „min_samples_split“ from scikit-learn. However, it has a 100 times bigger default value because two is too little for possibly infinite data streams and cannot significantly influence the distribution. By default, it has a value of 200, and it is pretty reasonable as the results from the Figure 6.10 present, as the best results, a provided by models with „grace_period“ between 200 and 300.

6.4.4 Split confidence

Split confidence expresses the allowed error rate, and the closer to 0 it is, the longer it takes the model to decide, but the better predictions it makes. The Figure 6.11 demonstrates that smaller values indeed provide better results. Nevertheless, assuming the fact that model training and classification time are essential for DDoS protection, the optimal remains the default value of 10^{-7}.

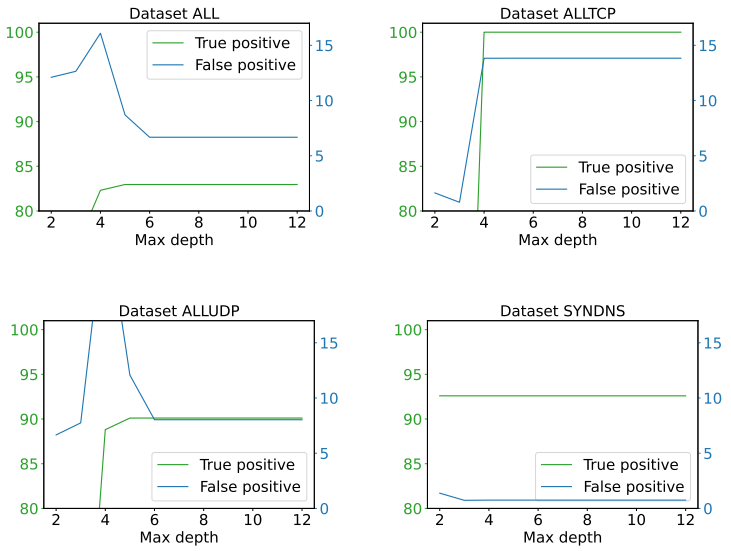


Figure 6.9: Hoeffding Adaptive Tree Classifier: maximum depth.

6.4.5 Tie threshold.

The last examined individual hyper-parameter „tie_threshold“ is a „*threshold below which a split will be forced to break ties*“[4]. When there are two feature candidates for splitting, which have similar values of information gain, it takes a lot of computational resources to decide the best one. When the „tie_threshold“ value is used as a comparing condition, the candidate node is split on the current best attribute [61]. So, the tree splits its nodes more often with smaller values of the tie threshold.

It is conspicuous that the tree model provides better results with a threshold value of 0,2 and greater (Figure 6.12), which is bigger than the default value of 0,05; so the leaf nodes do not split too often, which encourages pre-pruning and generalisation.

6.4.6 Grid search

Like for the Decision Tree Classifier, the aim was to show that a model can perform better if a suitable combination of parameters is found. Looking at the results in the Table 6.4, the True Positives rate in ALLUDP and ALL datasets catch the eye as they do not ever reach 90%. Moreover, the 5,66% of False Positives is far from a superb result. Entirely unexpected is the uniformity of the best parameter combinations. Again as in the case of the Decision Tree Classifier, maximum depth does not need to be restricted. The „grace_period“ and „tie_threshold“ hyper-parameters values coincide with the individual test results.

6.4.7 Mix to attack ratio

This part of the experiments test the incremental model's performance on different MIX to attack dataset ratio with the best-found combination of parameters. Compared to the Decision Tree Classifier, there is no clear tendency for the True Positive rate. On the contrary, the trends of False Positive values are almost identical to those which the offline

[4] https://riverml.xyz/latest/api/tree/HoeffdingAdaptiveTreeClassifier/

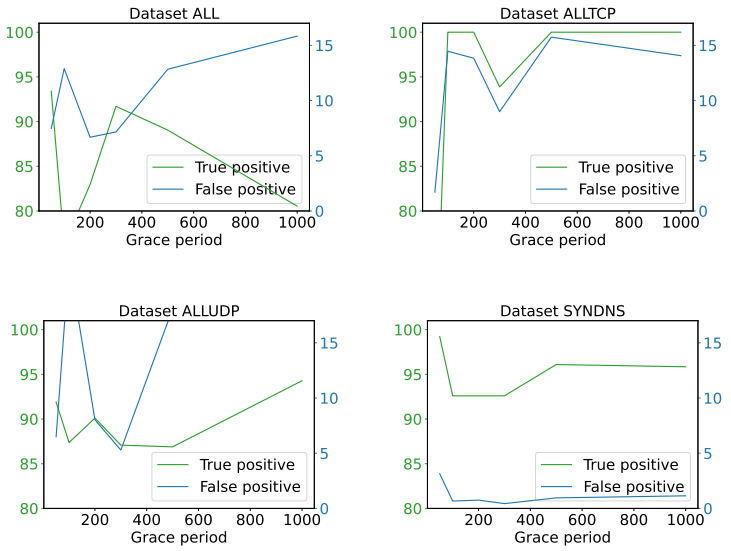


Figure 6.10: Hoeffding Adaptive Tree Classifier: grace period.

Dataset	**positives_ weight**	**max_ depth**	**grace_ period**	**split_ confidence**	**tie_ threshold**	**Results (TP, FP)**
SYNDNS	0,8	≥12	300	any	≥0,2	99,93%, 0,46%
ALLUDP	0,7; 0,8	≥12	300	any	≥0,2	89,53%, 3,95%
ALLTCP	0,7	≥12	400	any	≥0,2	100%, 2,53%
ALL	0,8	≥12	300	any	≥0,2	89,90%, 5,66%

Table 6.4: Grid-search results for Hoeffding Adaptive Tree Classifier for the individual datasets.

classifier has shown. False Positives results for ALLUDP and SYNDNS are about 10% and for ALL and ALLTCP are above 30% in the worst cases.

However, the erratic results of the True Positive rate, presented in Figure 6.13, demonstrate that the incremental decision tree cannot be used for the future application for DDoS defence, at least without the necessary optimisations.

6.4.8 Knapsack results for incremental learning

The „Legitimate rate“ with a 0/1 knapsack algorithm helped the Decision Tree classifier decrease the False Positive rate when the MIX to attack dataset rate was greater than 50%. A similar effect was expected in online learning on data streams. Nevertheless, it has not lived up to the expectations and demonstrates an extremely high False Positive rate even where the model did relatively good without the optimisation. Moreover, even worse are the results of the True Positive rate as Figure 6.14 illustrates. It concludes that providing additional information about the rate of attack samples in the stream does not improve the incremental model's performance and should be used only for offline learning.

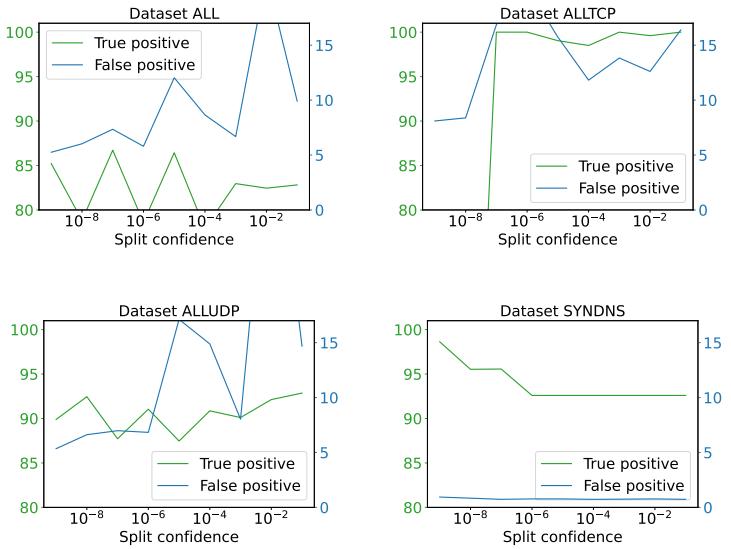


Figure 6.11: Hoeffding Adaptive Tree Classifier: split confidence.

6.5 Results Evaluation

The results of Decision Tree Classifier performance described in Section 6.3 conclude that the offline learning algorithm can be applied as a module for DDoS Protector. It has demonstrated a False Positive rate under 10% for all the datasets, where the attack rate was 60% or higher, even without the optimisation of the „Legitimate rate“ parameter (Figure 6.6). The optimisation helps to decrease the False Positives rate when the attack traffic is only a half or less of the incoming traffic, especially for ALL and ALLTCP datasets, as Figure 6.7 illustrates.

In comparison with the offline approach, the incremental learning algorithm performs worse. Figure 6.13 demonstrates unacceptable results of True Positive rate of Hoeffding Adaptive Tree Classifier algorithm, which are not only lower than provided by the offline approach but also absolutely unpredictable. Unfortunately, even the designed optimisations do not save the situation, as is visible in Figure 6.14.

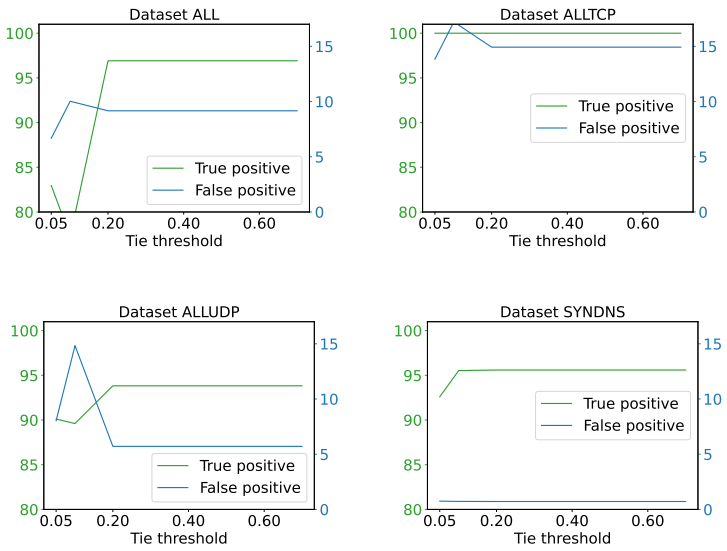


Figure 6.12: Hoeffding Adaptive Tree Classifier: tie threshold.

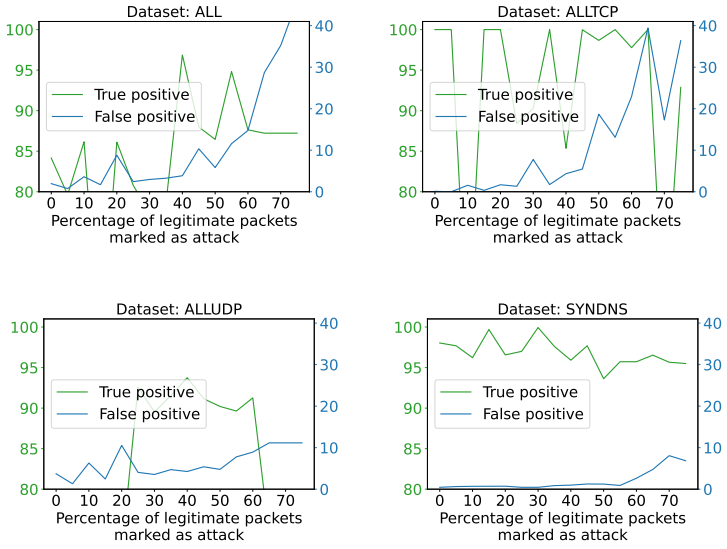


Figure 6.13: Hoeffding Adaptive Tree Classifier: percentage of legitimate packets marked as attack.

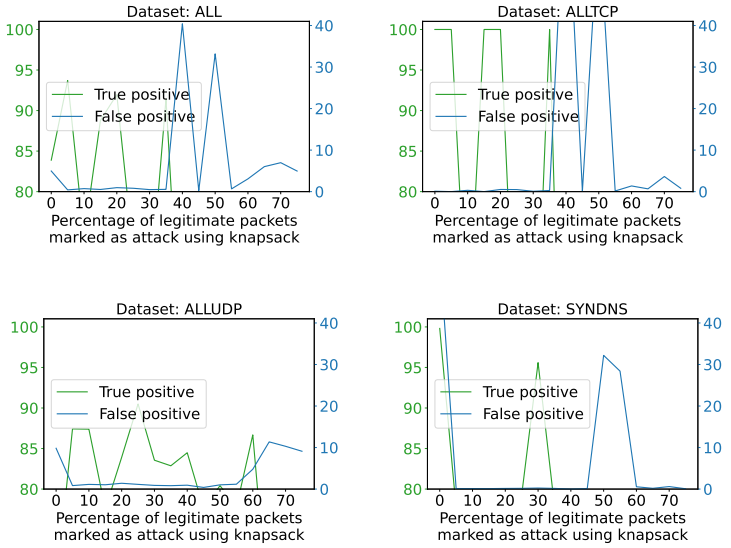


Figure 6.14: Hoeffding Adaptive Tree Classifier: percentage of legitimate packets marked as attack using 0/1 knapsack optimisation.

Chapter 7

Conclusions

This work's primary goal is to design an automatic traffic classifier for DDoS attack flows and optimisations. The decision trees were chosen because they belong to explainable machine learning algorithms, which are more trustworthy than opaque algorithms. Another reason is the ease of their output interpretability, which does not require any mathematical tools or a mathematical background from the end-users. A program using decision trees for offline supervised learning and online learning on data streams was implemented to infer filtration rules in BPF format, which are added to the database. As a bonus, the algorithm enables to obtain the output rules in Wireshark format.

One of the designed optimisations is grid search for finding the best hyper-parameters combination instead of tuning only one hyper-parameter and observing the tendencies of traffic classification of various models. Another optimisation is the introduction of the „Legitimate rate“ input parameter, which provides the algorithm information about the anticipated rate of legitimate packets in traffic, which aims to decrease the False Positive rate, i.e. the legitimate packets labelled as attack by the machine learning algorithm.

The incremental learning Hoeffding Adaptive Tree Classifier was supposed to benefit from changing attack vector detection and inference of blocking rules. Unfortunately, the inconsistent results of the algorithm learned on the data stream for one complex attack conclude that it cannot be used as an ML module for the DDoS protector. It is better to use the offline learning decision tree, which is not only more precise but also faster. The Decision Tree Classifier achieves above 98% of attack traffic recognition and less than 3% of False Positives even for relatively complex attack vectors, where the attack to legitimate traffic ratio is $3 : 7$. The „Legitimate rate“ optimisation using the 0/1 knapsack algorithm helps diminish the side effect of attack mitigation – blocking legitimate users – when attack flows take only 50% or less of the incoming traffic. This optimisation reduces the legitimate traffic blocking up to 10%, which is a noticeable improvement.

Even though the experiments with the Decision Tree Classifier and its optimisation have shown a satisfactory performance, there is still a space for future optimisations to minimise the rate of legitimate traffic blocked by the inferred rules. Until then, the output rules should be applied cautiously, and the future algorithm's update can be the implementation of the level of trustworthiness provided by the program. It would probably be based on the degree of similarity of attack traffic samples and the ratio of attack to legitimate traffic since the experiments show the best results when the rate of the attack traffic is 70% and above.

Bibliography

[1] Adadi, A. and Berrada, M. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*. 2018, vol. 6, p. 52138–52160. DOI: 10.1109/ACCESS.2018.2870052. ISSN 2169-3536. Available at: https://ieeexplore.ieee.org/document/8466590/.

[2] Belle, V. and Papantonis, I. Principles and Practice of Explainable Machine Learning. *Frontiers in Big Data*. 2021, vol. 4. DOI: 10.3389/fdata.2021.688969. ISSN 2624-909X. Available at: https://www.frontiersin.org/article/10.3389/fdata.2021.688969.

[3] Belyaev, M. and Gaivoronski, S. Towards load balancing in SDN-networks during DDoS-attacks. In: *2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC)*. Los Alamitos, CA, USA: IEEE Computer Society, Oct 2014, p. 1–6. DOI: 10.1109/MoNeTeC.2014.6995578. ISBN 978-1-4799-7595-2. Available at: https://doi.ieeecomputersociety.org/10.1109/MoNeTeC.2014.6995578.

[4] Bifet, A. and Gavaldà, R. Adaptive Learning from Evolving Data Streams. In:. August 2009, p. 249–260. DOI: 10.1007/978-3-642-03915-7_22. ISBN 978-3-642-03914-0.

[5] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A. et al. API design for machine learning software: experiences from the scikit-learn project. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, p. 108–122.

[6] Bussmann, N., Giudici, P., Marinelli, D. and Papenbrock, J. Explainable Machine Learning in Credit Risk Management. *Computational Economics*. January 2021, vol. 57, no. 1, p. 203–216. DOI: 10.1007/s10614-020-10042-0. Available at: https://ideas.repec.org/a/kap/compec/v57y2021i1d10.1007_s10614-020-10042-0.html.

[7] CESNET. *Adaptivní ochrana proti DDoS útokům* [online]. 2019 [cit. 2022-10-02]. Available at: https://www.cesnet.cz/projects/adaptivni-ochrana-proti-ddos-utokum/?lang=en.

[8] Chickowski, E. *Types of DDoS attacks explained* [online]. AT&T Cybersecurity, 8. july 2020 [cit. 2022-04-10]. Available at: https://cybersecurity.att.com/blogs/security-essentials/types-of-ddos-attacks-explained.

[9] Cloudflare, Inc.. *What is a Botnet?* [online]. 2017 [cit. 2022-03-09]. Available at: https://www.cloudflare.com/learning/ddos/what-is-a-ddos-botnet/.

[10] Cormen, T., Cormen, T., Leiserson, C., Books24x7, I., Technology, M. I. of et al. *Introduction To Algorithms*. MIT Press, 2001. Introduction to Algorithms. ISBN 9780262032933. Available at: https://books.google.cz/books?id=NLngYyWFl_YC.

[11] Fang, L., Wu, H., Qian, K., Wang, W. and Han, L. A Comprehensive Analysis of DDoS attacks based on DNS. *Journal of Physics: Conference Series*. 1st ed. IOP Publishing. sep 2021, vol. 2024, no. 1, p. 012027. DOI: 10.1088/1742-6596/2024/1/012027. Available at: https://doi.org/10.1088/1742-6596/2024/1/012027.

[12] Feinstein, L., Schnackenberg, D., Balupari, R. and Kindred, D. Statistical approaches to DDoS attack detection and response. In: *Proceedings DARPA Information Survivability Conference and Exposition*. 2003, vol. 1, p. 303–314 vol.1. DOI: 10.1109/DISCEX.2003.1194894.

[13] Fessi, B., Benabdallah, S., Boudriga, N. and Hamdi, M. A multi-attribute decision model for intrusion response system. *Information Sciences*. 2014, vol. 270, p. 237–254. DOI: https://doi.org/10.1016/j.ins.2014.02.139. ISSN 0020-0255. Available at: https://www.sciencedirect.com/science/article/pii/S0020025514002527.

[14] G, D., Rao, C., Singh, M. and Satyanarayana, G. A Survey on Defense Mechanisms countering DDoS Attacks in the Network. *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*. july 2013, vol. 2, p. 2599–2606.

[15] Gepperth, A. and Hammer, B. Incremental learning algorithms and applications. In: *European Symposium on Artificial Neural Networks (ESANN)*. Bruges, Belgium: ESANN, 2016. Available at: https://hal.archives-ouvertes.fr/hal-01418129.

[16] Gil, T. M. and Poletto, M. MULTOPS: A Data-Structure for Bandwidth Attack Detection. In: *10th USENIX Security Symposium (USENIX Security 01)*. Washington, D.C.: USENIX Association, August 2001. Available at: https://www.usenix.org/conference/10th-usenix-security-symposium/multops-data-structure-bandwidth-attack-detection.

[17] Hayes, T. L., Kafle, K., Shrestha, R., Acharya, M. and Kanan, C. REMIND Your Neural Network to Prevent Catastrophic Forgetting. In: Vedaldi, A., Bischof, H., Brox, T. and Frahm, J.-M., ed. *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, p. 466–483. DOI: 10.1007/978-3-030-58598-3_28. ISBN 978-3-030-58598-3.

[18] Holzinger, A. From machine learning to explainable AI. In: IEEE. *2018 world symposium on digital intelligence for systems and machines (DISA)*. 2018, p. 55–66. DOI: 10.1109/DISA.2018.8490530. ISBN 978-1-5386-5102-5.

[19] Huang, L., Huang, M., Guo, B. and Zhang, Z. A New Method for Constructing Decision Tree Based on Rough Set Theory. In: *2007 IEEE International Conference on Granular Computing (GRC 2007)*. IEEE, 2007, p. 241–241. DOI: 10.1109/GrC.2007.13. ISBN 0-7695-3032-X.

[20] Imperva. *DDoS attacks* [online]. Imperva, 2021 [cit. 2022-04-10]. Available at: https://www.imperva.com/learn/ddos/ddos-attacks/.

[21] Jacko, D. *Odvozování pravidel pro mitigaci DDoS útoků*. Brno, CZ, 2021. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Available at: https://www.fit.vut.cz/study/thesis/23920/.

[22] Jin, C., Wang, H. and Shin, K. G. Hop-Count Filtering: An Effective Defense against Spoofed DDoS Traffic. In: *Proceedings of the 10th ACM Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, 2003, p. 30–41. CCS '03. DOI: 10.1145/948109.948116. ISBN 1581137389. Available at: https://doi.org/10.1145/948109.948116.

[23] Jordan, M. I. and Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science*. 2015, vol. 349, no. 6245, p. 255–260. DOI: 10.1126/science.aaa8415. Available at: https://www.science.org/doi/abs/10.1126/science.aaa8415.

[24] Karunanidhi, K. ARROS: Distributed Adaptive Real-Time Network Intrusion Response. In:. 2006.

[25] Kekely, L., Cabal, J., Puš, V. and Kořenek, J. Multi Buses: Theory and Practical Considerations of Data Bus Width Scaling in FPGAs. In: *Proceedings - Euromicro Conference on Digital System Design, DSD 2020*. IEEE Computer Society, 2020, p. 49–56. DOI: 10.1109/DSD51259.2020.00020. ISBN 978-1-7281-9535-3. Available at: https://www.fit.vut.cz/research/publication/12341.

[26] Kholidy, H. A., Erradi, A., Abdelwahed, S. and Baiardi, F. A Risk Mitigation Approach for Autonomous Cloud Intrusion Response System. *Computing*. Berlin, Heidelberg: Springer-Verlag. nov 2016, vol. 98, no. 11, p. 1111–1135. DOI: 10.1007/s00607-016-0495-8. ISSN 0010-485X. Available at: https://doi.org/10.1007/s00607-016-0495-8.

[27] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G. et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*. 2017, vol. 114, no. 13, p. 3521–3526. DOI: 10.1073/pnas.1611835114. Available at: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

[28] Kotsiantis, S. B. Decision trees: a recent overview. *Artificial Intelligence Review*. 2011, vol. 39, p. 261–283.

[29] Krämer, L., Krupp, J., Makita, D., Nishizoe, T., Koide, T. et al. AmpPot: Monitoring and Defending Against Amplification DDoS Attacks. In: Bos, H., Monrose, F. and Blanc, G., ed. *Research in Attacks, Intrusions, and Defenses*. Cham: Springer International Publishing, 2015, p. 615–636. DOI: 10.1007/978-3-319-26362-5_28. ISBN 978-3-319-26362-5.

[30] Li, J., Liu, Y. and Gu, L. DDoS attack detection based on neural network. In: *2010 2nd International Symposium on Aware Computing*. 2010, p. 196–199. DOI: 10.1109/ISAC.2010.5670479.

[31] Lin, S.-C. and Tseng, S.-S. Constructing detection knowledge for DDoS intrusion tolerance. *Expert Systems with Applications*. 2004, vol. 27, no. 3, p. 379–390. DOI: https://doi.org/10.1016/j.eswa.2004.05.016. ISSN 0957-4174. Available at: https://www.sciencedirect.com/science/article/pii/S0957417404000417.

[32] Luo, Y., Yin, L., Bai, W. and Mao, K. An Appraisal of Incremental Learning Methods. *Entropy*. 2020, vol. 22, no. 11. DOI: 10.3390/e22111190. ISSN 1099-4300. Available at: https://www.mdpi.com/1099-4300/22/11/1190.

[33] Lutes, J. *Entropy and Information Gain in Decision Trees* [online]. 2020. Available at: https://towardsdatascience.com/entropy-and-information-gain-in-decision-trees-c7db67a3a293.

[34] Lyon, R., Brooke, J., Knowles, J. and Stappers, B. Hellinger Distance Trees for Imbalanced Streams. *Proceedings - International Conference on Pattern Recognition*. may 2014. DOI: 10.1109/ICPR.2014.344.

[35] Ma, X. and Chen, Y. DDoS Detection Method Based on Chaos Analysis of Network Traffic Entropy. *IEEE Communications Letters*. 2014, vol. 18, no. 1, p. 114–117. DOI: 10.1109/LCOMM.2013.112613.132275.

[36] Mahajan, D. and Sachdeva, M. DDoS Attack Prevention and Mitigation Techniques - A Review. *International Journal of Computer Applications*. 1st ed. april 2013, vol. 67, no. 19, p. 21–24. DOI: 10.5120/11504-7221.

[37] Mahjabin, T., Xiao, Y., Sun, G. and Jiang, W. A survey of distributed denial-of-service attack, prevention, and mitigation techniques. *International Journal of Distributed Sensor Networks*. 1st ed. december 2017, vol. 13, no. 12. DOI: 10.1177/1550147717741463.

[38] Malialis, K., Devlin, S. and Kudenko, D. Distributed reinforcement learning for adaptive and robust network intrusion response. *Connection Science*. Taylor & Francis. 2015, vol. 27, no. 3, p. 234–252. DOI: 10.1080/09540091.2015.1031082. Available at: https://doi.org/10.1080/09540091.2015.1031082.

[39] Minegishi, T. and Niimi, A. Detection of fraud use of credit card by extended VFDT. In:. March 2011, p. 152 – 159. DOI: 10.1109/WorldCIS17046.2011.5749902.

[40] Mirkovic, J. and Reiher, P. D-WARD: a source-end defense against flooding denial-of-service attacks. *IEEE Transactions on Dependable and Secure Computing*. 2005, vol. 2, no. 3, p. 216–232. DOI: 10.1109/TDSC.2005.35.

[41] Mohri, M., Rostamizadeh, A. and Talwalkar, A. *Foundations of Machine Learning*. 2nd ed. Cambridge, MA: MIT Press, 2018. Adaptive Computation and Machine Learning. ISBN 978-0-262-03940-6.

[42] Obaid, H. Denial of Service Attacks: Tools and Categories. *International Journal of Engineering Research and Technology*. 1st ed. IJERT. april 2020, V9, no. 03. DOI: 10.17577/IJERTV9IS030289.

[43] Palmer, D. *DDoS attacks are cheaper and easier to carry out than ever before* [online]. ZDNet, 11. november 2020 [cit. 2022-04-10]. Available at:

https://www.zdnet.com/article/ddos-attacks-are-cheaper-and-easier-to-carry-out-than-ever-before/.

[44] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. and Wermter, S. Continual lifelong learning with neural networks: A review. *Neural Networks*. 2019, vol. 113, p. 54–71. DOI: https://doi.org/10.1016/j.neunet.2019.01.012. ISSN 0893-6080. Available at: https://www.sciencedirect.com/science/article/pii/S0893608019300231.

[45] Pei, J., Chen, Y. and Ji, W. A DDoS Attack Detection Method Based on Machine Learning. *Journal of Physics: Conference Series*. IOP Publishing. jun 2019, vol. 1237, no. 3, p. 032040. DOI: 10.1088/1742-6596/1237/3/032040. Available at: https://doi.org/10.1088/1742-6596/1237/3/032040.

[46] Peng, T., Leckie, C. and Ramamohanarao, K. Protection from distributed denial of service attacks using history-based IP filtering. *IEEE International Conference on Communications, 2003. ICC '03.* 2003, vol. 1, p. 482–486 vol.1.

[47] Podgorelec, V., Kokol, P., Stiglic, B. and Rozman, I. Decision Trees: An Overview and Their Use in Medicine. *Journal of medical systems*. november 2002, vol. 26, p. 445–63. DOI: 10.1023/A:1016409317640.

[48] Quinlan, J. Decision trees and decision-making. *IEEE Transactions on Systems, Man, and Cybernetics*. 1990, vol. 20, no. 2, p. 339–346. DOI: 10.1109/21.52545.

[49] Ragsdale, D., Carver, C., Humphries, J. and Pooch, U. Adaptation techniques for intrusion detection and intrusion response systems. In: *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0.* 2000, vol. 4, p. 2344–2349 vol.4. DOI: 10.1109/ICSMC.2000.884341.

[50] Read, J., Bifet, A., Pfahringer, B. and Holmes, G. Batch-Incremental versus Instance-Incremental Learning in Dynamic and Evolving Data. In: Hollmén, J., Klawonn, F. and Tucker, A., ed. *Advances in Intelligent Data Analysis XI*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, p. 313–323. DOI: 10.1007/978-3-642-34156-4_29. ISBN 978-3-642-34156-4.

[51] ROBINS, A. Catastrophic Forgetting, Rehearsal and Pseudorehearsal. *Connection Science*. Taylor & Francis. 1995, vol. 7, no. 2, p. 123–146. DOI: 10.1080/09540099550039318. Available at: https://doi.org/10.1080/09540099550039318.

[52] Seeliger, A., Pfaff, M. and Krcmar, H. Semantic web technologies for explainable machine learning models: A literature review. *PROFILES/SEMEX@ ISWC*. 2019, vol. 2465, p. 1–16.

[53] Shabtai, A., Kanonov, U. and Elovici, Y. Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method. *Journal of Systems and Software*. 2010, vol. 83, no. 8, p. 1524–1537. DOI: https://doi.org/10.1016/j.jss.2010.03.046. ISSN 0164-1212. Performance Evaluation

and Optimization of Ubiquitous Computing and Networked Systems. Available at: https://www.sciencedirect.com/science/article/pii/S0164121210000762.

[54] Sharafaldin, I., Lashkari, A. H., Hakak, S. and Ghorbani, A. A. Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy. In: *2019 International Carnahan Conference on Security Technology (ICCST)*. 2019, p. 1–8. DOI: 10.1109/CCST.2019.8888419.

[55] Stewart, M. *The Limitations of Machine Learning*. 2019. Available at: https://towardsdatascience.com/the-limitations-of-machine-learning-a00e0c3040c6.

[56] The Hacker News. *Reasons Why Every Business is a Target of DDoS Attacks* [online]. 2022 [cit. 2022-03-09]. Available at: https://thehackernews.com/2022/01/reasons-why-every-business-is-target-of.html.

[57] Toklu, S. and Şimşek, M. Two-Layer Approach for Mixed High-Rate and Low-Rate Distributed Denial of Service (DDoS) Attack Detection and Filtering. *ARABIAN JOURNAL FOR SCIENCE AND ENGINEERING*. april 2018, vol. 43, p. 7923–7931. DOI: 10.1007/s13369-018-3236-9.

[58] Tyagi, N. *Understanding the Gini Index and Information Gain in Decision Trees* [online]. 2020. Available at: https://medium.com/analytics-steps/understanding-the-gini-index-and-information-gain-in-decision-trees-ab4720518ba8.

[59] Vishwakarma, R. and Jain, A. K. A Honeypot with Machine Learning based Detection Framework for defending IoT based Botnet DDoS Attacks. In: *2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, April 2019, p. 1019–1024. DOI: 10.1109/ICOEI.2019.8862720. ISBN 978-1-5386-9439-8.

[60] Yaar, A., Perrig, A. and Song, D. StackPi: New Packet Marking and Filtering Mechanisms for DDoS and IP Spoofing Defense. *IEEE Journal on Selected Areas in Communications*. 2006, vol. 24, no. 10, p. 1853–1863. DOI: 10.1109/JSAC.2006.877138.

[61] Yang, H. and Fong, S. Moderated VFDT in Stream Mining Using Adaptive Tie Threshold and Incremental Pruning. In: Cuzzocrea, A. and Dayal, U., ed. *Data Warehousing and Knowledge Discovery*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, p. 471–483. DOI: 10.1007/978-3-642-23544-3_36. ISBN 978-3-642-23544-3.

[62] Yusof, M. A. M., Ali, F. H. M. and Darus, M. Y. Detection and Defense Algorithms of Different Types of DDoS Attacks. *International Journal of Engineering and Technology*. 1st ed. october 2017, vol. 9, no. 5, p. 410–414. DOI: 10.7763/IJET.2017.V9.1008.