Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Linnea Fossum Gustavsen

Alert Fusion: A Combined Approach for DDoS Attack Detection

Master's thesis in Communication Technology
Supervisor: Yuming Jiang
Co-supervisor: Otto Wittner, Sikt
July 2023

NTNU
Norwegian University of Science and Technology

Linnea Fossum Gustavsen

Alert Fusion: A Combined Approach for DDoS Attack Detection

Master's thesis in Communication Technology
Supervisor: Yuming Jiang
Co-supervisor: Otto Wittner, Sikt
July 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Title: Alert Fusion: A Combined Approach for DDoS Attack Detection

Student: Gustavsen, Linnea Fossum

Problem description:

Distributed Denial of Service (DDoS) attacks are becoming more prevalent in today's networks. This increase in attack traffic challenges Internet Service Providers (ISPs) to protect their networks. Normally, an ISP collects a lot of monitoring data to manage the network and analyze for security incidents. However, the results of the analyses are typically looked at separately, which may cause some attacks to go undetected.

To help DDoS detection, different data sources can be analyzed to provide a comprehensive overview of the characteristics of the traffic. The results of their anomaly analyses can be combined and correlated to provide a more holistic view of the attack situation. As such, this holistic view gives the security team a better chance at detecting DDoS attacks in the network.

This project will investigate using different traffic anomaly detection methods to produce alerts of anomalies on data sources that are related but may be of different types. Such methods include thresholds based on empirical statistical models and entropy measures. A correlation technique will be proposed to put the produced alerts in context. The performance of the correlation method will be evaluated on datasets available from the ISP Sikt.

Approved on: 2023-02-17
Main supervisor: Professor Jiang, Yuming, NTNU
Co-supervisor: Wittner, Otto, Sikt

Abstract

Distributed Denial of Service (DDoS) attacks are evolving in size and sophistication as technology progresses, posing significant threats to both Internet access customers and providers. Internet Service Providers (ISPs) can employ Intrusion Detection Systems (IDSs) to detect such attacks. However, as attacks become more sophisticated, IDSs must also evolve accordingly. One potential approach to enhance IDS capabilities is through collaborative methods. This thesis explores the effectiveness of utilizing different anomaly detection methods and combining them in an IDS. The exploration follows a design process based on the design science research method.

An IDS is implemented with 52 methods and an alert fusion component to investigate the impact of various detection methods and their combination. These methods utilize NetFlow records or router interface telemetry measurements to detect 11 different types of DDoS attacks. Evaluation of method effectiveness relies on ground truth data from three rounds of generated attacks within a real ISP network. While specific methods demonstrate promise in detecting DDoS attacks within specific environments, most methods do not exhibit clear effectiveness.

Through the combination of individual detection method outputs, the alert fusion component of the IDS successfully detects 10 out of the 11 possible attacks. Additionally, the combined alerts are summarized and ranked based on their perceived level of danger. The effectiveness of the alert fusion approach varies depending on the user's definition of dangerous events and their criteria for a precise combination of alerts. Fusion methods are employed based on time, attack type, IP address, and packet size distribution. Time-based fusion yields the poorest results among these fusion approaches, with only a slight improvement in precision compared to the incoming alerts, dependent on different parameters. In contrast, fusion based on packet size distribution exhibits a high alert reduction and improved precision.

In conclusion, this thesis reveals that the efficacy of various anomaly detection methods in detecting DDoS attacks is heavily influenced by multiple factors, limiting the ability to generalize about their performance. However, combining these methods can yield additional benefits, depending upon parameters chosen during the implementation of the individual detection methods and the user's definition of benefits.

Sammendrag

Distribuerte tjenestenektangrep blir stadig større og mer avanserte i takt med teknologisk utvikling, og utgjør en trusel både for kunder og leverandører av internett-tilgang. Internetleverandører kan oppdage angrep ved å implementere innbruddsdeteksjonssystemer i nettverket sitt, men ettersom angrepene blir mer avanserte, bør også systemene bli det. En måte å gjøre dette på kan være å utnytte fordelene av samarbeid. Denne avhandlingen undersøker effekten av å bruke ulike metoder for avviksdeteksjon og kombinere resultatene deres. Undersøkelsen gjennomføres ved å følge en designprosess basert på forskningsmetoden "design science".

Et innbruddsdeteksjonssystem er implementert med 52 metoder og en varslingssammenslåingskomponent for å undersøke effekten av ulike deteksjonsmetoder og deres kombinasjon. Disse metodene bruker NetFlow data eller målinger fra rutergrensesnitt for å oppdage 11 forskjellige typer distribuerte tjenestenektangrep. Evalueringen av metodenes effektivitet baserer seg på trafikk data fra tre runder med genererte angrep i et reelt internetleverandør-nettverk. Mens enkelte metoder viser lovende resultater ved oppdagelse av angrep i spesifikke omstendigheter, viser de fleste metodene ingen tydelig effektivitet.

Ved å kombinere resultatene fra de individuelle deteksjonsmetodene, oppdager varslingssammenslåingskomponenten 10 av de 11 mulige angrepene. I tillegg oppsummeres og rangeres de kombinerte varslene basert på oppfattet faregrad. Effektiviteten av varslingssammenslåingen varierer avhengig av brukerens definisjon av farlige hendelser og kriteriene for en presis kombinasjon av varsler. Sammenslåingsmetoder basert på tid, angrepstype, IP-adresse og distribusjon av pakkestørrelser benyttes. Tidsbasert sammenslåing gir dårligst resultater, med bare en liten forbedring i nøyaktighet sammenlignet med innkommende varsler, avhengig av ulike parametere. Derimot viser sammenslåing basert på distribusjon av pakkestørrelser en betydelig reduksjon i antall varsler og forbedret nøyaktighet.

Konklusjonen av denne avhandlingen er at effektiviteten til ulike deteksjonsmetoder i stor grad påvirkes av mange faktorer, noe som begrenser muligheten for å si noe generelt om deres ytelse. Å kombinere disse metodene kan gi ytterligere fordeler, avhengig av parameterne som velges under implementeringen av de individuelle deteksjonsmetodene og brukerens definisjon av fordeler.

Preface

This thesis concludes my academic journey as a student pursuing a Master of Science in Communication Technology at the Norwegian University of Science and Technology (NTNU). The research project for my master's thesis was conducted in collaboration with the Internet Service Provider (ISP) Sikt. The main objective of the thesis is to explore the effect of different anomaly detection methods in detecting Distributed Denial of Service (DDoS) attacks and if the combination of these methods provides additional benefits in the detection.

This topic sparked my interest because it combines my passion for networking and security. Throughout the work on this thesis, this passion has grown. Finishing this thesis is in no small part due to the immense help from my supervisors, professor Yuming Jiang and Otto J. Wittner (Sikt). With their continuing support and great insights, the thesis took shape. I also want to thank Emil Henry Flakk (Sikt) for providing me with the NetFlow data and the machine to calculate my detections on. Lastly, I would like to thank the wonderful staff and students at the Department of Information Security and Communication Technology for providing a great and safe learning environment during my five years at NTNU.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AI Artificial Intelligence.

AUC area under curve.

CERT Computer Emergency Response Team.

CPU Central Processing Unit.

DDoS Distributed Denial of Service.

DNS Domain Name System.

DPI Deep Packet Inspection.

FFT Fast Fourier Transform.

FNR False Negative Rate.

FPR False Positive Rate.

GUI graphical user interface.

HTTP Hypertext Transfer Protocol.

ICMP Internet Control Message Protocol.

IDS Intrusion Detection System.

IoT Internet of Things.

IP Internet Protocol.

ISP Internet Service Provider.

IXP Internet Exchange Point.

kNN k-Nearest Neighbors.

LAN Local Area Network.

LRDDoS Low-Rate Distributed Denial of Service.

NREN National research and education network.

PPV Positive Predictive Value.

RF Random Forest.

ROC receiver operating characteristic.

rps requests per second.

R.U.D.Y R U Dead Yet?.

SNMP Simple Network Management Protocol.

TCP Transmission Control Protocol.

TNR True Negative Rate.

TPR True Positive Rate.

UDP User Datagram Protocol.

VM Virtual Machine.

Chapter 1
Introduction

1.1 Motivation

As attackers are getting smarter, cybersecurity professionals have to work harder. Distributed Denial of Service (DDoS) attacks have become more prevalent and destructive, posing a significant risk to the availability of essential services [1]. Availability is a fundamental aspect of secure services, as the sudden unavailability of critical infrastructure can have severe consequences. Industry standards dictate that services should be available 99.999% of the time. However, DDoS attacks threaten this standard by overloading services, rendering them unavailable to legitimate users. In the first quarter of 2023, a record-breaking volumetric Hypertext Transfer Protocol (HTTP) DDoS attack occurred, reaching a rate of 71 million requests per second (rps) [1].

The increase of Internet of Things (IoT) devices has made it easier for attackers to assemble botnets, distributed sources from which attacks are launched. Often characterized by poor security standards, these devices are often exploitable, making them ideal for botnet recruitment [2]. The ease of launching DDoS attacks has contributed to their increasing prevalence. Attackers can purchase access to botnets, granting them substantial attack power [3]. Unlike many other attacks, DDoS attacks do not necessarily rely on exploiting vulnerabilities but rather overwhelm the target with packets. Furthermore, these attacks continue to grow in scale each year [4], stressing the urgency of countermeasures.

Over the past two decades, hackers have used DDoS attacks as a tool for various motivations [5]. Some attackers seek financial gain, holding services ransom until a payment is made. In contrast, others engage in activism, exemplified by groups like the pro-Russian hacking group Killnet or the famous hacking group Anonymous. A notable instance of a recent DDoS attack occurred on the 26th of June 2022, when Killnet targeted multiple Norwegian services, disrupting essential everyday services. Even the website of the Norwegian National Security Authority fell victim to this

attack [6].

As technology advances, so are DDoS attacks. In the first quarter of 2023, Domain Name System (DNS)-based attacks emerged as the most prevalent type of network-layer DDoS attacks [1]. One prominent example is the DNS reflection amplification attack, which uses the DNS protocol. This attack involves the DNS server receiving requests that generate large response packets, with the attacker spoofing their Internet Protocol (IP) address to redirect the amplified response to the victim. DNS amplification reflection attacks require fewer resources than other flooding attacks, as a relatively small packet can result in a much larger packet reaching the victim. Such attacks have seen a significant increase in prevalence, rising by 958% since the previous quarter [1].

Attackers are also employing increasingly sophisticated techniques to evade detection. Exploiting vulnerabilities in targeted applications, they render these applications unavailable. Launching attacks at low or slow rates minimizes bandwidth consumption, reducing the chances of triggering alarms. Additionally, attacks exploiting vulnerabilities are harder to detect since they appear as seemingly regular packets, causing substantial damage to the victim [7]. Unveiling the attackers' intentions, in this case, requires examining the packet payload, which is challenging due to the prevalence of encrypted traffic and the location of some detection systems. Attackers may also enhance their stealth by combining different attacks or protocols to mask one attack with another or confuse detection mechanisms [8].

DDoS attacks also threaten the companies providing Internet access. In the first quarter of 2023, the telecommunications industry ranked fifth in terms of HTTP DDoS attacks, making Internet Service Providers (ISPs) high-level targets [1]. Furthermore, in network-layer attacks like those operating at lower levels in the protocol stack, the telecommunications industry ranked third in attack frequency.

These circumstances necessitate companies deploying intelligent Intrusion Detection Systems (IDSs). IDSs play a crucial role in detecting potential attacks, including DDoS attacks. The effectiveness of an IDS in detecting different attacks depends on its location and the employed detection methods. Host-based IDSs can more easily detect sophisticated application-layer attacks as they have access to decrypted packets.

Implementing an IDS in small to medium-sized ISPs presents unique challenges. Commercial IDS solutions are often expensive, making it difficult for ISPs to justify the investment. Consequently, ISPs may resort to manual traffic monitoring, analyzing collected data and inspecting individual results, or adopting open-source IDS software. However, manually analyzing results is time-consuming and may result in oversight, leading to undetected attacks. Employing a single detection method may prove

ineffective due to the variations in attack patterns. A one-size-fits-all approach may not apply in such cases. If an ISP solely relies on an IDS that detects known attacks, unknown attacks and slight variations of known attacks may go unnoticed. To effectively detect sophisticated attacks, an IDS should employ multiple detection methods and combine their outputs [9]. The combination of multiple detection methods can provide better performance due to the redundancy it gives [10]. Each detection method can provide complementary information about potential attack patterns [11].

1.2 Contributions

This thesis contributes five important factors:

1. a comprehensive combination of detection methods
2. an approach to fuse the results from these methods
3. results on the individual detection methods and from the fusion
4. open source code to implement this system[1]
5. validation of the results using ground truth from real attacks.

By combining multiple different detection methods, the proposed approach can achieve better performance compared to individual methods [12]. This combination can reduce False Positive Rate (FPR) and improve accuracy [9]. Given the diverse range of detection methods, the proposed approach can cover a broader collection of attack types, potentially detecting more attacks. The outputs from different detection methods provide a holistic understanding of the potential attack situation, assisting network managers in gaining a comprehensive overview and facilitating effective attack mitigation [9].

Fusing the outputs of the detection methods enables a comprehensive view of the security landscape. While individual outputs may offer limited insights, their combination paints a complete picture, providing a global perspective on the security situation [13].

Each individual detection method produces its own results, as does the combination of these methods. These results offer insights into the detection methods' performance independently and collectively. The performance is validated using data collected during a period known to include attacks generated for this thesis.

[1] https://github.com/linneagustavsen/master-thesis

1.3 Research questions

Based on the motivation behind this thesis, the following research questions were formulated:

R1 What is the effect of different anomaly detection methods when used to detect DDoS attacks?

R2 Does the combination of these methods give additional benefits in attack detection, and if so, how?

1.4 Outline of thesis

Following this introductory chapter, the background information necessary for understanding the content of this master's thesis will be presented in Chapter 2. Subsequently, a review of relevant literature about the topics addressed in this thesis will be presented in Chapter 3. Chapter 4 will outline the process employed to obtain the results discussed in Chapter 6. The proposed attack detection method is described in Chapter 5. Chapter 6 presents how the real attacks were generated, which were needed to validate the performance of the proposed method. This chapter also presents the outputs and performance of all parts of the proposed IDS. Lastly, a conclusion is made in Chapter 7.

Chapter 2
Background

This chapter will provide the essential background information to understand the rest of the thesis. This chapter will introduce different types of DDoS attacks, methods to detect them, and various data sources for such detection purposes.

2.1 DDoS attacks

A DDoS attack is a malicious act aimed at rendering a target unavailable. For example, if the target is a web server providing a service, the attacker aims to disrupt legitimate user access by launching a DDoS attack [14].

Researchers have tried classifying different types of DDoS attacks. One of the most commonly cited taxonomies was proposed by Jelena Mirkovic and Peter Reiher in 2004 [15]. The classification has been supplemented using additional taxonomies due to the outdated nature of some parts of their taxonomy. In most academic papers, DDoS attacks are classified based on the specific weakness they exploit to achieve their objectives. Mirkovic and Reiher refer to these categories as vulnerability and flooding attacks [15].

2.1.1 Flooding attacks

Flooding attacks exploit the network resources of the victim, which could encompass the switching capabilities and routers' buffers [16]. These attacks involve the transmission of an extensive volume of traffic toward the targeted victim. As a consequence of the overwhelming traffic, the victim network becomes depleted, rendering it incapable of attending to legitimate traffic. Flooding attacks are often easier to detect in monitoring data from routers, as they consume significant network resources [16].

There are several well-known types of flooding attacks, including Internet Control Message Protocol (ICMP) flood and User Datagram Protocol (UDP) flood [16]. An

ICMP flood entails the attacker bombarding the victim network with ICMP echo requests, commonly known as pings. Network administrators typically utilize ping messages to assess the responsiveness of services within their network [17]. Network managers may block ICMP packets to mitigate this attack.

On the other hand, a UDP flood involves bombarding the victim network with UDP datagrams targeted at random ports of a victim host [18]. Unlike Transmission Control Protocol (TCP) datagrams, UDP datagrams do not require a three-way handshake, as UDP does not establish a connection. The connectionless nature of UDP makes it particularly suitable for flooding attacks, as the attacker can initiate the transmission of a significant volume of packets without setting up a connection first [18].

2.1.2 Vulnerability attacks

Vulnerability attacks exploit carefully crafted packets that exploit weaknesses in the communication protocols at the application, transport, and network layers or in the targeted service. By consuming the resources of the victim host, these packets prevent the host from handling legitimate traffic. Due to their particular construction, vulnerability attacks may not require a flood of attack traffic, making them difficult to detect.

One widely recognized vulnerability attack is the TCP SYN flood, which exploits the three-way handshake procedure TCP utilizes to establish connections. An attacker will try to establish many connections simultaneously but never finish the handshake. This causes the server to maintain a large number of half-open connections. As the server's capacity to track half-open connections becomes exhausted, it becomes unable to handle legitimate handshake requests, thus achieving the objective of the attack.

Two additional examples of vulnerability attacks are the Xmas flooding attack and the Blacknurse attack. The Xmas flooding attack derives its name from the flood of TCP packets in which all flags are set to one, causing the packet to "light up" like a Christmas tree. This peculiar behavior confuses the victim and significantly strains its resources as it struggles with processing these anomalous packets. On the other hand, the Blacknurse attack involves the transmission of ICMP destination unreachable packets [19]. This attack does not necessitate a high bandwidth to achieve success as it leaves firewalls with a high Central Processing Unit (CPU) load, rendering them incapable of handling legitimate traffic [19].

One type of vulnerability attack that demonstrates sophistication is the Low-Rate Distributed Denial of Service (LRDDoS) attack. LRDDoS attacks employ a strategy of transmitting seemingly legitimate traffic at a deliberately low rate to

evade detection [20]. By reducing the packet rate below the detection thresholds, LRDDoS attacks effectively conceal the malicious traffic within benign network traffic [21]. Several examples of LRDDoS attacks include SlowLoris, Slow Read, and R U Dead Yet? (R.U.D.Y). SlowLoris uses the application layer protocol HTTP to establish an overwhelming number of connections between the attacker and a victim server without closing them [22]. This tactic results in the exhaustion of the victim's resources, leading to unavailability.

It is worth noting that Apache HTTP servers using default settings prior to version 2.3.15 are susceptible to SlowLoris attacks [23], which affects over 13% of all websites utilizing Apache HTTP web servers [24]. Disturbingly, Shodan[1] reveals the existence of over 1 million web servers currently employing versions vulnerable to SlowLoris attacks [25].

Another attack that exploits the application layer protocol HTTP is the Slow Read attack. This attack involves sending a genuine request to a web server but deliberately reading the page at an extremely slow pace, thereby keeping the connection open for an extended duration [26]. This will preoccupy connections that could be used for legitimate clients. Like the SlowLoris attack, Slow Read affects Apache HTTP servers utilizing default settings before version 2.3.15.

R.U.D.Y DDoS attacks, also operating using the HTTP protocol, targets explicitly the HTTP method POST. Instead of a slow reading approach, R.U.D.Y posts data to the victim server very slowly. The attacker initiates a POST request indicating their intention to submit a lengthy message. Subsequently, the attacker transmits this message incrementally in small fragments with extended intervals between each transmission. The attacker consumes resources that could otherwise be utilized for legitimate traffic by prolonging the connection for an extensive period. The web server may become unavailable if the attacker launches multiple slow post connections. Like SlowLoris and Slow Read, mitigating R.U.D.Y involves configuring the timeout for receiving requests to a reasonably low time interval [23].

2.2 Attack generation

Various methods can generate DDoS attacks, often involving free-to-use tools that enable attackers to initiate these attacks with a simple button press. To amplify the damage of such attacks, attackers commonly leverage the utilization of a botnet, which refers to a collection of infected devices under the control of an adversary known as a botmaster [27]. The botmaster will direct the devices in the botnet to execute specific actions. The power gained from controlling numerous devices can give a DDoS attack tremendous strength. Often, IoT devices become compromised

[1] A search engine for internet-connected services

and enlisted as part of botnets due to their inherent vulnerabilities, such as the utilization of default credentials [2]. Certain services even offer the option to pay for access to the capabilities of a botnet.

The widely-used Linux distribution Kali Linux provides tools for launching DDoS attacks, including hping3 [28] and slowhttptest [29]. Hping3 facilitates the generation of custom ICMP, UDP, or TCP packets and sends them at different speeds. The highest speed setting, known as "flood", involves sending packets as rapidly as possible. On the other hand, slowhttptest enables the execution of sophisticated LRDDoS attacks such as Slowloris, Slow Read, and R.U.D.Y.

Understanding the complexities of attack generation facilitates the detection of such attacks. In the case of flooding attacks, the attacker generates them by bombarding the victim with a large volume of traffic. This activity manifests in the network traffic as a surge of traffic directed toward a specific IP address. Additionally, the traffic often disproportionately represents the flooding attack's protocol.

Vulnerability attacks exhibit distinct characteristics depending on the specific attack type. The behavior of the three-way handshake can be focused on when considering the TCP SYN flood attack. Under normal circumstances, a client initiates a TCP connection by transmitting a packet to the server with the SYN flag set. The server responds with a packet with the SYN and ACK flags set. If the client is a legitimate user, it concludes the handshake procedure by sending a packet to the server with the ACK flag set, thereby establishing the connection. However, in a TCP SYN flooding attack, the client intentionally omits the final ACK packet and instead sends a packet with only the SYN flag set. By excluding the last ACK, the attacker prevents the establishment of the connection. Consequently, a high volume of SYN packets destined for the same destination can indicate the occurrence of such an attack.

The Xmas attack involves utilizing TCP packets with an unusual combination of flags. Due to the abnormality arising from the simultaneous setting of all flags, this attack can be relatively straightforward to detect. The Blacknurse attack entails the transmission of ICMP packets of type 3 (Destination Unreachable) and code 3 (Port Unreachable) [19]. This attack increases the frequency of packets with this particular type, which observers can notice within the traffic.

Detecting LRDDoS attacks is challenging since they employ legitimate HTTP packets that blend in with benign traffic [30]. However, these attacks may still increase the packet count directed toward the victim's IP address, which observers can identify within the traffic.

2.3 Attack detection

Network managers commonly employ an IDS within a network to identify various DDoS attacks. The primary function of an IDS is to continuously monitor the network and promptly notify network managers of potential intrusions. The design and configuration of an IDS can vary regarding its placement and the methods employed for detection. The specific configuration of an IDS depends on the intended scope of protection and the specific types of attacks it aims to mitigate.

2.3.1 Measures of performance

To evaluate the performance of an IDS, a range of metrics are available for computation. An IDS can classify a generated alert into one of the following four categories: False Positive (FP), False Negative (FN), True Positive (TP), or True Negative (TN). A False Positive refers to an event flagged as an intrusion by the IDS, even though it is benign. Contrarily, a False Negative occurs when a genuinely malicious event is not flagged as an intrusion by the IDS. On the other hand, a True Positive represents an event correctly identified as an intrusion by the IDS, indicating its malicious nature. Lastly, a True Negative refers to an event accurately recognized as non-malicious and not flagged as an intrusion by the IDS.

Once these terms have been established, several metrics can be computed to evaluate the performance of the IDS. One widely utilized metric is accuracy, seen in Equation (2.1). This metric measures the percentage of events correctly categorized by the IDS.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

Another metric is the FPR, which can be seen in Equation (2.2). The FPR measures the percentage of benign events classified as malicious.

$$FPR = \frac{FP}{FP + TN} \tag{2.2}$$

False Negative Rate (FNR) measures the percentage of malicious events classified as benign, seen in Equation (2.3).

$$FNR = \frac{FN}{FN + TP} \tag{2.3}$$

Precision, also known as Positive Predictive Value (PPV), measures the percentage of the alerts that are correctly classified as malicious events [30], seen in Equation (2.4).

$$Precision = \frac{TP}{TP + FP} \tag{2.4}$$

True Positive Rate (TPR), also called recall, measures the percentage of malicious events classified as malicious, seen in Equation 2.5.

$$TPR = \frac{TP}{TP + FN} \tag{2.5}$$

The F1-score is the trade-off between precision and recall [31]. It is a way to evaluate precision and recall in a single metric. It is designed to work well on imbalanced data, e.g., where the true negatives outweigh the true positives. The F1-score is calculated by Equation (2.6).

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{2.6}$$

Using the TPR and FPR, a curve called the receiver operating characteristic (ROC) curve can be plotted. These rates are graphically represented by plotting them against each other, with separate axes assigned to each rate. The resulting curve is determined by the varying TPR and FPR values obtained for a given detection method, depending on the threshold utilized to differentiate between positive and negative instances.

Similarly, the precision-recall curve can be plotted with precision and recall on separate axes. This curve is used to visualize the trade-off between having precise alerts and detecting all attacks. The visualization enables an analysis of the precision and recall values using various thresholds.

The area under both the ROC and precision-recall curves can be found, quantifying these visualizations. This area is often referred to as area under curve (AUC) and is in the range $[0, 1]$. An AUC of 0.5 for the ROC curve indicates that whether or not the detection method can distinguish between positive and negative instances is random.

When tuning an IDS, a fundamental trade-off exists between false positives and false negatives [30]. This trade-off arises because achieving fewer false positives requires setting a higher threshold for triggering alerts. However, this higher threshold increases the likelihood of overlooking genuine alerts, potentially leading to a higher rate of false negatives.

2.3.2 Location

An IDS can be positioned in various locations depending on the specific detection objectives. Two commonly considered logical placements are within a host system or the network. Host-based IDSs typically possess access to internal host-related artifacts, such as logs, user activities, and program execution patterns. This access proves valuable when detecting entities like malware or unauthorized agents. On the

other hand, network-based IDSs operate by monitoring the network traffic passing through them. Placing an IDS at the entry point of a Local Area Network (LAN) enables it to block malicious traffic from infiltrating the network. In the context of this thesis, the focus will be on network-based IDSs. This choice stems from the constraint that an ISP cannot install IDSs on the customer hosts of its network.

2.3.3 Detection method

An IDS utilizes two primary categories of detection methods: misuse detection, also called signature-based detection, and anomaly-based detection [30]. Misuse detection relies on a collection of predefined signatures that represent known attack patterns. It tries to match incoming traffic against these signatures to detect attacks. These signatures are typically in a packet-based format, which is not the typical type of data an ISP collects. To obtain packet data with payloads, the ISP must employ Deep Packet Inspection (DPI) techniques. However, given the prevalence of encryption in modern network traffic, DPI would yield less valuable information for an ISP. Consequently, anomaly-based detection is better suited for an ISP network.

Anomaly detection

When employing anomaly-based detection, a model representing the normal state of the network is constructed, and deviations from this model are considered potential intrusions [30]. This approach enables the detection of unknown attacks but also entails a higher FPR than misuse detection. The higher FPR comes from the fact that there might be a lot more deviations from the "normal" network state that is legitimate traffic than when a packet matches with an attack signature. Consequently, an inherent trade-off exists between the FPR and the FNR. If the detection sensitivity is too high, many false positives may be generated while minimizing false negatives. Conversely, reducing false positives might increase the risk of missing subtle attacks, resulting in higher false negatives.

An anomaly-based IDS operates under the assumption that malicious behavior differs from benign behavior [30]. However, this assumption may only sometimes hold, particularly in scenarios like LRDDoS attacks where the traffic pattern closely resembles benign traffic. An effective anomaly-based IDS should be capable of detecting these subtle nuances. Numerous methods, including statistical techniques and machine learning, can be employed to construct the model representing normal network behavior.

2.3.3.1 Statistical methods

Statistical methods can detect network attacks using numerical distributions to model network behavior. Unlike machine learning methods, most statistical approaches do

not require prior knowledge about normal or attack behaviors as they learn from the incoming data. By comparing incoming measurements to previous measurements, statistical methods excel at detecting sudden attacks [30]. Moreover, they can provide a deviation score indicating the extent to which new measurements deviate from the previous ones, a feature not intuitively offered by machine learning techniques. However, statistical methods may not always be optimal for detecting gradual or minor attacks [30].

Anomaly detection using statistical methods encompasses various techniques, including information theory metrics such as entropy, entropy rate, and information distance. Information theory metrics are employed to quantify the impact of new observations [32]. Entropy, a fundamental concept in information theory, measures the level of uncertainty or randomness within a system. A higher entropy value indicates greater randomness, while a lower entropy value signifies more certainty or predictability [33]. It can be used for anomaly detection because a sudden increase or decrease in the randomness of a network feature can be considered an anomaly. Entropy as an information metric was first proposed by Claude Shannon in his paper "A Mathematical Theory of Communication", published in 1948 [33]. He defines the entropy by Equation (2.7), where X is a random variable and $p_1, ..., p_n$ are the probabilities in the probability distribution $P(X)$.

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \tag{2.7}$$

The maximum entropy value is attained when all probabilities, denoted as p_i, are equal to $\frac{1}{n}$. This signifies that all events are equally likely, resulting in the highest uncertainty surrounding the information. Suppose this is applied to anomaly detection where X is the probability distribution of all destination IP addresses within a time interval. The entropy of the probability distribution of destination IP addresses can provide insights into the level of certainty regarding the distribution. During a DDoS attack, there might be an increase of packets directed towards a specific destination IP address, potentially indicating the victim of the attack. This deviation from the expected distribution would manifest as a decrease in entropy since there is a strengthened certainty that this particular IP address is the destination for the packets.

In 2011, Yang Xiang, Ke Li, and Wanlei Zhou proposed an approach for calculating the generalized entropy in their paper titled "Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics" [34]. Equation (2.8), where X is a random variable, $p_1, ..., p_n$ are the probabilities in the probability distribution $P(X)$ and α is the order of the entropy.

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \left(\sum_{i=1}^{n} p_i^\alpha \right) \tag{2.8}$$

When α is one, the generalized entropy converges to Shannon's entropy. The advantage of using this metric in anomaly detection is that the higher α is, the higher the deviation between different probability distributions becomes [34]. This can help to differentiate between benign and anomalous traffic. Another metric the paper by Yang Xiang, Ke Li, and Wanlei Zhou [34] proposed was information distance, as defined in Equation (2.9) where P and Q are two probability distributions, and α is the order.

$$D_\alpha(P,Q) = \frac{1}{\alpha - 1} \log_2 \left(\sum_{i=1}^{n} p_i^\alpha q_i^{1-\alpha} \times \sum_{i=1}^{n} q_i^\alpha p_i^{1-\alpha} \right) \tag{2.9}$$

The information distance metric measures how similar or different P and Q are to one another. The authors use it to detect if the probability distribution of benign traffic P differs enough from the presumed attack distribution Q.

Other statistical methodologies include volume-based thresholds, statistical modeling, and analysis of a histogram's top k bins. Volume-based thresholds can be employed for anomaly detection to identify significant increases in network traffic. For instance, a threshold can be set on the number of packets transmitted through a router, and surpassing this threshold would indicate an anomaly. Statistical modeling allows for identifying attacks by creating a model based on normal network conditions. Subsequently, any substantial deviations of incoming data from the established model would be considered anomalous. The last statistical approach involves examining the traffic within a specific interval by constructing a histogram. This histogram, such as one displaying the number of packets per destination IP address, enables observing sudden address changes. Suppose a new destination IP address appears among the top k addresses regarding packet count. In that case, it may indicate a previously unseen host receiving significant traffic, implying an anomaly.

2.3.3.2 Machine learning

Machine learning for anomaly detection has gained significant prominence as technology continues to evolve. Machine learning, a subset of Artificial Intelligence (AI), encompasses algorithms that can learn patterns and establish connections within data [35]. Machine learning methods can be categorized into three distinct types: supervised, unsupervised, and semi-supervised learning. The distinguishing factor between these methods is how they learn the patterns and connections.

Supervised learning methods involve training machine learning models using labeled data, where the algorithm is provided both the raw data and the corresponding ground truth. During training, the model attempts to learn the associations between the raw data combinations and their corresponding ground truth labels. This acquired knowledge enables the model to apply this learned behavior to unlabeled data. In contrast, unsupervised learning methods do not require labeled data to identify

patterns. Instead, they examine the similarities and dissimilarities present in the raw data to generate groupings accordingly. On the other hand, semi-supervised learning leverages a small portion of labeled data alongside a more significant portion of unlabeled data for training purposes. Using a small set of labeled data proves advantageous when labeling data is expensive.

Machine learning algorithms are helpful in anomaly detection due to their ability to uncover hidden patterns within network traffic data. However, these algorithms exhibit high sensitivity to the training data they were exposed to and the specific features they were provided [30]. Supervised learning algorithms, especially, face limitations in detecting unknown attacks since they rely on training data that includes only a set of known attack types.

A widely adopted supervised machine learning approach is known as Random Forest (RF), which enables the classification of observations within a given data set. The underlying principle of the RF method involves instantiating multiple decision trees that collectively contribute to classifying incoming data [36]. Decision trees, another supervised machine learning algorithm, split the incoming data based on conditions deduced from the feature set. The aim is to achieve a perfect split that aligns with the classification classes. During the training phase of a decision tree, it aims to identify the optimal data partitioning strategy to reach a perfect separation. It does this by maximizing the information gain computed from the entropy of the split. A lower entropy value indicates higher information gain because it is less uncertain. Within the RF framework, decision trees are trained on subsets of the data, employing different subsets of features to mitigate the risk of overfitting the model.

Once the forest of decision trees is trained, it can classify new measurements. When a new data point traverses a decision tree, it reaches a leaf node associated with a specific classification label, enabling the decision tree to make a corresponding announcement. The RF classifier aggregates the outcomes the individual decision trees announced, with the majority consensus serving as the final output. The strength of the RF method lies in its utilization of multiple decision trees, enabling the model to compensate for potential errors made by individual trees.

An example of a prevalent unsupervised machine learning technique is K-means clustering. The objective of this method is to partition a set of measurements into k clusters based on their similarity, thereby grouping similar measurements [37]. Central to K-means clustering is cluster centroids, representing the mean of the measurements within each cluster; hence the “means” in K-means. Initially, during the clustering process, the cluster centroids are randomly assigned, and the incoming measurements are allocated to their appropriate centroids. Once the clustering

process has grouped the measurements, the algorithm recalculates the centroids of the clusters. Subsequently, the data points within each cluster are compared to the new centroids to determine if any measurements should be re-assigned to another cluster. This iterative process continues until the cluster centroids stabilize or until a predefined iteration threshold has passed, signifying the completion of the clustering procedure. The time required for clustering to stabilize depends on the initial values of the cluster centroids; poorly chosen initial centroids can extend the convergence process as data points struggle to find their appropriate clusters. To apply K-means clustering to anomaly detection, it becomes necessary to identify which cluster or clusters are anomalous. One general assumption is that the biggest cluster represents normal behavior; however, this assumption may not hold in the case of large flooding attacks. In such instances, clustering metrics can be calculated to get more information about the structure of the clusters. These metrics help to determine what kind of data are in the different clusters.

2.3.3.3 Alert fusion

A detection method has the potential to generate a significant volume of alerts. When multiple detection methods are employed, the number of alerts further escalates. This situation can become overwhelming for the security team responsible for, e.g., the network. To mitigate this challenge, a method known as alert fusion can be employed to reduce the number of alerts.

Alert fusion is the procedure of combining the alerts from multiple sources to get a more meaningful and intuitive result [38]. The procedure consists of four steps to reduce the number of alerts and give a more holistic view of the situation. The first step is to normalize the alerts. Normalization aims to get the alerts from the different detection methods in the same format, which is crucial for combining them. The second step is to aggregate the alerts. Aggregation aims to group alerts that are close in time and have similar features [39]. This step fuses alerts with the same view of the situation. The third step is correlating the alerts. Alert correlation aims to cluster similar alerts and discard irrelevant alerts to obtain a holistic view of the possible attack situation [9]. Dingbang Xu and Peng Ning [40] propose four techniques to correlate alerts:

1. Similarity between alert attributes,
2. Predefined attack scenarios,
3. Prerequisites and consequences of attacks,
4. Multiple information sources

and a fifth one; filtering algorithms, is proposed by Elshoush and Osman [12]. Correlating based on the similarity between alert attributes can be applied by, e.g., clustering alerts with the same IP addresses or port numbers. This is useful if an attack is launched against a single destination IP address. Then all of the alerts

with this destination IP address could correspond to the same attack, and multiple alerts would not be needed. How to define similarity is also extensively researched. "Similarity" could mean identical fields of the alerts like the same IP address, or it could be defined by calculating some value to measure the similarity between two alerts.

A predefined attack scenario could be that a port scan comes before a buffer overflow [40]. This is much easier in signature-based IDSs because the signatures can be used in the scenario. The technique based on prerequisites and consequences of attacks is similar to attack scenarios. The difference is that here, the attack scenario is built up by matching the consequences of an earlier attack to the prerequisites of the current attack [12]. An example could be that the consequence of a port scan is that an open port is found, and the prerequisite of a buffer overflow is that a port is open. This is also easier in a signature-based IDS.

Multiple information sources mean that there are multiple heterogeneous sources generating alerts. To correlate these alerts, there needs to be some preexisting knowledge about the architecture of the system the different detection methods are monitoring. In a network-based IDS, this may be the network topology and, e.g., its subnets. Filtering algorithms are used to remove irrelevant alerts. Alerts not relevant to the monitored system can, e.g., be discarded.

The fourth step is to rank the alerts. The ranking could be based on different variables, but the goal is to represent the alerts to the security team responsible for the network so that they can act accordingly. Alerts considered more dangerous should be at the top of the ranking.

2.4 Data sources

An IDS needs data sources to detect intrusions effectively. The choice of data sources varies depending on the detection method and location within the network. In the case of a network-based IDS, the logical data source is traffic data, which can be in either a packet-based or a flow-based format. Packet-based traffic data encompasses capturing every packet that traverses a designated collection point. The granular nature of packet-based traffic makes it easier for an IDS to detect all the attacks without missing them [30]. However, due to its fine-grained nature, packet-based data scales poorly in bigger networks.

Consequently, it is common practice to aggregate traffic data on a flow basis, where similar traffic is reduced into flows. Flows contain essential header information from the aggregated packets, such as source and destination IP addresses, ports, and other relevant details. The aggregation of packets into flows enhances scalability in

large networks, although it still incurs significant storage overhead [30].

An alternative form of traffic data aggregation, less frequently employed as a data source for an IDS, is router interface-based data. This data type is typically employed for network management but provides an aggregated representation of the overall traffic situation. Within the scope of this thesis, the focus will primarily revolve around two specific data sets, NetFlow and router interface telemetry.

2.4.1 NetFlow

NetFlow, an extensively adopted flow-based collection protocol, was developed by Cisco in the late 1990s [41]. It facilitates the collection of network packets and organizes them into unidirectional flows based on the presence of shared key fields. These key fields include source and destination IP addresses, source and destination ports, and the transport layer protocol or type of service [42]. When referring to NetFlow data, it means the flow records that contain information about these unidirectional flows [43].

Aggregating flows before transmission to the network manager introduces a slight delay, potentially enabling an attack to succeed before detection [30]. However, since NetFlow data contain packet header fields, such as IP addresses, it helps trace the source or target of an attack, a capability that interface-based traffic data lacks. To manage the cost associated with collecting NetFlow data, network managers often implement a technique called sampling. Sampling involves aggregating packets into flows at a specific frequency. The sampling frequency may also decrease as costs increase. This approach balances preserving the packet stream structure while minimizing storage and computational expenses [44]. Nevertheless, it is essential to note that sampling inevitably results in some loss of information, thereby inhibiting a comprehensive view of the entire traffic landscape.

2.4.2 Telemetry

Telemetry is the term for collecting measurements from remote sensors transmitted to a collection point. In a network setting, this can apply to router interface-based measurements. In the context of this thesis, telemetry data specifically refers to the measurements derived from the router interfaces within the network. A sensor deployed on a router interface can capture various metrics, including the volume of incoming and outgoing bytes and packets or the queue size. Traditionally, this information is used for monitoring network performance. However, during a flooding DDoS attack, the network's performance may change significantly, rendering telemetry data useful for intrusion detection. The traffic load on the network is better shown in the telemetry measurement than in the NetFlow flow records because of the sampling.

Consequently, these measurements may be more valuable when detecting substantial traffic surges.

Chapter 3
Literature on detection methods and related work

Numerous studies have explored the detection of DDoS attacks, resulting in much research. To prepare for the work in this thesis, an exhaustive review of the relevant literature was conducted. This review resulted in a categorization of DDoS attack detection methods into statistical methods and machine learning techniques. Additionally, the focal point of this thesis, namely, alert fusion was investigated. The subsequent section will provide an overview of the findings obtained from the literature study.

3.1 Literature on detection methods

3.1.1 Statistical methods

One widely employed statistical method in research involves the utilization of information theory metrics, such as entropy, information distance, and information gain. Xiang et al. [34] introduced two novel information metrics for detecting LRDDoS attacks, namely generalized entropy and a new distance metric. According to their claims, the utilization of generalized entropy, as opposed to Shannon's entropy, yields a more noticeable difference in entropy between regular and attack traffic, thereby resulting in a higher detection rate. Additionally, Xiang et al. [34] claim that their metrics can reduce FPR by up to 199% compared to Shannon's entropy. While no specific calculations were provided to support this assertion, another study [21] reported an F1 score of 0.50 and 0.71 for LRDDoS attacks based on two distinct data sets using their method.

Another research paper that uses entropy to detect DDoS attacks is Tsobdjou et al. [45]. They use Shannon's entropy to detect attacks in a five-module system. Initially, the researchers extract features, such as the source and destination IP addresses and the packet timestamp, from the incoming network packets. With these features, they create bi-directional connections during a specific period. The entropy component processes these connections and their corresponding packet counts in

each direction. In this context, the researchers consider the source IP address as the random variable, and they determine the probability of a specific source IP address based on the proportion of packets originating from that address. They calculate the entropy based on these probabilities and report suspicious activity when the calculated entropy falls below a predetermined threshold. Subsequently, module three is triggered when suspicious activity is detected, identifying the attack connections. They define an attack connection as a connection where the combination of the number of packets in both directions exceeds a threshold. The alert generation module is activated when there is a report of an attack connection. It is worth noting that the thresholds are dynamically adjusted and updated over time. However, the claim made in the paper regarding 100% accuracy and a 0% FPR seems suspicious.

Feinstein et al. [46] incorporate entropy and Chi-Squared analysis to detect DDoS attacks by examining frequency-sorted distributions of selected packet features. An illustrative example of such a packet feature is the IP address, which facilitates generating a distribution representing unique source IP addresses over a specific sampling period. The researchers compare the generated distribution with other sampling periods by calculating the entropy. When the feature measurements are discrete, like protocol or flags set, Chi-Squared analysis is employed to compare the distributions. It is important to note that Feinstein et al. [46] acknowledge the potential challenges the Chi-Squared detector faces when differentiating between legitimate and attack traffic during stealthy attack scenarios.

Information metrics are not the only statistical method for detecting DDoS attacks. Sekar et al. [47] introduce a triggered, multi-stage detection architecture for an IDS explicitly designed for large-scale networks. They specifically tailor their IDS to operate efficiently in high-traffic networks, such as ISP's core networks. The scalability of their approach stems from the triggered methodology, which comprises two detection stages. The first stage entails lightweight anomaly detection, employing Simple Network Management Protocol (SNMP) data feeds to trigger the subsequent stage that utilizes NetFlow data. A trigger is sent from the lightweight detection if there are any anomalies in traffic volume or link utilization. They detect anomalies by training a statistical model on historical SNMP data. They construct the model using actual SNMP data, with unknown anomalies eliminated using Fast Fourier Transform (FFT). The model assumes that the data exhibits a weekly pattern, implying that each week should display a similar pattern. Leveraging this assumption, the mean and variance are computed for the same time on the same weekday for k number of weeks. This will give the assumed normal weekly traffic pattern. Subsequently, this model calculates a deviation score, indicating the number of standard deviations by which a new measurement deviates from the model. If the deviation score surpasses a predefined threshold, the first stage sends a trigger to the second stage, which involves focused anomaly detection. In this stage, data are clustered based on IP

prefixes from multiple flow record sets. To evaluate the effectiveness of their detection system, the researchers compared it to a commercial DDoS detection system. The reported performance metrics include a FNR of 1.25% and a TPR of 84%.

Simple thresholds can also serve as effective means to detect DDoS attacks. In their work, David and Thomas [48] utilize dynamic thresholds based on traffic features. They establish their thresholds by relying on four assumptions regarding traffic behavior during DDoS attacks. These assumptions encompass that if there is a considerable number of unique source IP addresses, destination IP will be the same for each packet, potentially random destination ports, a high volume of packets, uniform protocol usage, and packet lengths ranging from 40 to 60 bytes. However, another article [8] criticizes these assumptions as being excessively rigid and possibly lacking applicability in real attack scenarios. As a result of their research, David and Thomas achieved an accuracy of 99.5%, a TPR of 99.5%, and a True Negative Rate (TNR) of 99.5%.

3.1.2 Machine learning

As technology advances, machine learning algorithms for detecting DDoS attacks are becoming more prevalent in research. The proposed method employs a RF classifier and K-means clustering as the chosen machine-learning techniques. Researchers have extensively investigated these methods, and a brief overview of relevant studies is presented below.

Supervised learning

Corrêa et al. [49] employ telemetry data from cloud Virtual Machines (VMs) to detect DDoS attacks. They make their data set from an experimental test bed where they launch TCP SYN flood and HTTP GET flood attacks. They conduct feature selection using Chi-Squared to identify a subset of k features that optimize accuracy when using the supervised learning method k-Nearest Neighbors (kNN). When the features were selected, they evaluated the performance of six machine learning methods, namely kNN, Naive Bayes, Support Vector Machine, Decision Tree, RF, and Multi-Layer Perceptron. The findings reveal that kNN and RF perform best. They both achieve accuracy and TPR exceeding 87%, a precision score surpassing 86%, and an F1 score exceeding 85%.

Monge et al. [50] conducted a study to detect a device's participation in a DDoS flooding attack in 5G networks. Their research employed a RF model trained on various flow features. The authors reported a TPR of 0.93 and a FPR of 0.01. However, their method exhibits two limitations. Firstly, the accuracy of the results diminishes when the time interval between flow captures exceeds 15 seconds. Secondly,

the precision of the RF model decreases when trained against a broader range of attacks.

Yungaicela-Naula et al. [51] investigated the performance of seven machine and deep learning algorithms in detecting transport-layer and application-layer attacks. The seven algorithms evaluated were three machine learning algorithms: kNN, Support Vector Machine, and RF, and four deep learning models: Multi-Layer Perceptron, Convolutional Neural Network, Gated Recurrent Units, and Long Short-Term Memory neural network. The transport-layer attacks considered were TCP SYN flood and UDP flood. The application-layer attacks included SlowLoris, R.U.D.Y, Slow read, HTTP GET flood, and Distributed and reflected DNS amplification attacks. The researchers conducted tests on two publicly available data sets, yielding promising outcomes, particularly for RF, which achieved over 96% accuracy in binary classification[1] with an FPR of less than 1% on one of the data sets.

Unsupervised learning

Tuan et al. [52] conducted a comprehensive evaluation of various machine learning algorithms to assess their effectiveness in detecting Botnet DDoS attacks. The algorithms under consideration included Support Vector Machine, Artificial Neural Networks, Naive Bayes, Decision Tree, and K-means. The researchers employed an open-source packet-based dataset for testing these algorithms. Their analysis revealed that K-means was the most successful in detecting Botnet DDoS attacks, achieving the lowest FPR and the highest accuracy. However, it is necessary to note that the FPR was alarmingly high, surpassing 90% for nearly all the evaluated algorithms, indicating suboptimal performance. K-means was the only algorithm exhibiting a FPR below 90%.

Liang and Znati [53] proposed a taxonomy of machine learning-based detection schemes for DDoS attacks to identify the most effective approaches. The authors compared the performance of several algorithms, including Decision Tree, Support Vector Machine, kNN, K-means, Naive Bayes, and Artificial Neural Networks. Their analysis did not identify a single method that outperformed all others, but K-means demonstrated promising results specifically in the context of TCP flows.

Mazel et al. [54] proposed an unsupervised approach for detecting anomalies, including DDoS attacks, by leveraging multiple independent analyses on flow-based data. The authors aimed to establish correlations among the results obtained from different aggregations applied to the same dataset, such as aggregating flows with the same IP or subnet. The analyses employes simple metrics, such as byte and packet counts, to identify abrupt changes in the data. Once the system detects an

[1] i.e., distinguishing between attack and benign traffic

abrupt change, it inputs it into an unsupervised learning algorithm for clustering. The clustering output is then correlated based on similarity and ranked according to the severity of the alerts, with particular emphasis on flooding attacks, considered the most dangerous. The proposed method showcased promising results regarding accuracy, but there is room for improvement in reducing the execution time of the clustering process.

3.1.3 Combination of multiple methods

The combination of multiple methods for detecting DDoS attacks has been an area of research interest. Zhou et al. [21] introduced a novel approach using five statistical features to identify DDoS attacks, including those composed of multiple protocols. These five features encompass the entropy rate of source IP address flows, entropy rate of bi-directional flows, entropy of packet sizes, entropy rate of packet sizes, and the count of ICMP destination unreachable packets. The authors employed Decision Tree, Deep Learning, kNN, Logistic Regression, RF, and Support Vector Machine classifiers to assess the performance of the proposed features. The results indicated that the RF classifier yielded the highest performance in binary classification for specific DDoS attacks. In contrast, for DDoS attacks comprising multiple protocols, RF exhibited superior performance in multi-class classification. Additionally, the authors compared their feature selection approach using RF with other state-of-the-art methods. Remarkably, the proposed feature selection achieved F1 scores of either 0.99 or 1 for all types of attacks present in the datasets, showcasing its effectiveness.

Another research paper that uses statistical methods and machine learning is Liu et al. [8]. They explored applying statistical methods and machine learning techniques to detect large-scale DDoS attacks, including those involving multiple protocols. Their approach involved training a neural network using features they get from extracting the "work of the stream data" and calculating the FFT coefficients and entropy of this work. The definition of the "work of the stream data" is: "the effect of stream data on the communication capacity" [8]. The stream data describes the network behavior, which implies that the work will be the behavior's impact on the network. They use the FFT coefficients and entropy of this work as features to train the neural network. The proposed method achieved a detection accuracy of over 99% for selected attack types, including TCP SYN attacks. However, the study had certain limitations. It could not detect zero-day attacks or attacks that conceal themselves within benign traffic, such as HTTP floods.

3.2 Related work on alert fusion

There are many ways to fuse data from numerous IDSs, and this has been researched extensively. Gu et al. [11] introduced the utilization of the likelihood ratio test for

alert fusion in their research. The likelihood ratio test was employed to establish fusion rules to minimize the system cost or identify the optimal trade-off between the FPR and FNR. They calculated the expected cost by assessing the cost associated with increasing false positives or negatives. Additionally, the authors suggested that the score generated from the likelihood ratio test could be employed to rank the fused alerts.

IDSs that use alert fusion have been developed before. An open source DDoS application called NeMo was developed by the German Computer Emergency Response Team (CERT) DFN-CERT [55]. The software is designed explicitly for National research and education networks (NRENs). This software uses metrics such as the number of SYNs, number of ACKs, TCP packets per second, etc., to detect DDoS attacks based on either a threshold or more advanced methods such as holt-winters seasonal forecasting. Alerts from different routers and metrics can be combined into something they call Meta Alerts. When a Meta Alert is manually created, incoming alerts that share the same affected router can be merged into this Meta Alert if the user chooses this. The Meta Alert will then include the aggregated alerts that might be connected to the same attack. They created Meta Alerts to reduce the messages the user gets. Although this open-source software is similar to the proposed method in this thesis, a notable difference is that NeMo does not fuse the alerts automatically, which the proposed method does.

3.2.1 Aggregation

The aggregation of alerts plays a crucial role in alert fusion. Maggi et al. [56] presented a method for aggregating "close in time" alerts using fuzzy measures and fuzzy sets. Their work contributes to establishing a time-distance criterion to determine if alerts should be considered "close in time", thus mitigating measurement errors. This approach's underlying idea is that alerts about the same event may be generated with slight time delays. Aggregating these alerts, despite having slightly different timestamps, can be beneficial. To implement their fuzzy measure, they introduced a short time window preceding the start time of an alert, within which the occurrence of the alert is considered possible. Consequently, if another alert falls within this time window, the two alerts are combined.

Yao et al. [57] propose an approach to aggregating alerts utilizing the principles of rough set theory. Their methodology involves assigning weights to each feature based on its significance in classifying attacks, such as DDoS attacks or port scans. They use the feature weights to calculate the similarity between pairs of features within each attack category. Additionally, they incorporate background information regarding the system on which the IDS is deployed to determine the reliability of the alert, taking into account known vulnerabilities. By employing these techniques, Yao

et al. achieved a reduction of over 80% in repetitive alerts.

Similarly, Wang et al. [10] present a method that utilizes a multi-layer feed-forward neural network to fuse alerts generated by heterogeneous sensors. Specifically, they employ the widely adopted host-based IDS Snort and NetFlow records as their sensor inputs. The multi-layer feed-forward neural network emulates the information processing mechanism of the human brain, comprising multiple neurons responsible for processing input data. The primary objective of their neural network is to learn the optimal approach for solving a given problem, in this case, classifying alerts into one of four attack types or normal behavior. Following the classification process, Wang et al. aggregate similar alerts based on time, attack type, and source IP address. The output of their approach is a vector that encapsulates the number of aggregated attacks, the time window within which the alerts were aggregated, and an indication of the severity of the alert aggregation.

3.2.2 Correlation

Hoque et al. [58] propose a multi-step approach to reduce false alerts and identify attack scenarios specifically for DDoS attacks. Their method consists of the following key stages. Firstly, they first filter away false alerts by referencing a knowledge base of known false alerts. If the system does not find an alert in the knowledge base, it computes the similarity score based on the number of similar alerts in the current and previous time window. An alert surpassing a predefined threshold of similarity is considered an actual alert and subsequently subjected to further correlation analysis. In the correlation stage, the alerts are first clustered based on their similarities. Then these clustered alerts are fused into a new alert representing the combined information from the individual alerts within each cluster. The final step involves correlating the fused alerts to construct an attack scenario. The method proposed by Hoque et al. has demonstrated significant efficacy in reducing the overall number of alerts generated.

This thesis took inspiration from some of these related research studies and applied them to the specific environment relevant for an ISP of a NREN. Detection methods inspired by the literature explained in Section 3.1 were implemented and combined using alert fusion. The alert fusion shares similarities with the research approaches in Section 3.2, like using a “weight” feature that will determine the likelihood that an alert is a true positive.

Chapter 4
Methodology

This chapter explores the methodology employed to address the research questions introduced in Section 1.3. The subsequent sections will outline the procedural steps to obtain the desired results.

An IDS was developed to address the research questions. A design process was followed to create this IDS, adhering to the principles of the design science research method. The definition of design science can be found in Roel Wieringa's book, "Design Science Methodology for Information Systems and Software Engineering" [59], seen in Definition 4.1.

Definition 4.1. Design science is the design and investigation of artifacts in context.

The artifact in this thesis is then the IDS, and the context is the incident response team or security department using this IDS and the network itself. The design science process aims to create an artifact that enhances specific aspects within the given context [59]. This thesis aims to develop an IDS that improves the ease and effectiveness of detecting DDoS attacks for the incident response team or security department. Following the research questions, the design problem entailed designing an IDS that detects DDoS attacks using multiple data sources and detection methods. The design cycle outlined in [59] is followed to achieve this. The design cycle consists of three steps: problem investigation, treatment design, and treatment validation.

4.1 Problem investigation

During the problem investigation phase, an extensive literature study was conducted on DDoS detection. The initial findings were presented in the specialization project report [60], which identified numerous relevant research papers. Throughout the thesis work, the literature study was continuously updated, aligning with the iterative nature of the design cycle. Valuable insights were derived from the literature study, providing various approaches for tackling the design problem. These included different detection methods and strategies for combining the outputs of these methods.

The subsequent step involved determining the appropriate data sources for detecting attacks. Many research studies primarily utilized NetFlow data, as it encompasses IP addresses and other valuable features for detection. This data set was included in the collection of data sources. Additionally, router interface telemetry data was included, as it offers a more granular view of traffic volume than the sampled NetFlow data. This finer-grained perspective could prove beneficial in detecting flooding attacks and potentially other attack types visible from traffic volume data.

Once the data sources were finalized, suitable detection methods were selected. Since the chosen data sources included network traffic data devoid of payload information, anomaly detection was the logical choice. Inspiration was drawn from three research papers, namely [47], [49], and [34], which guided the selection of detection methods. Additional methods were incorporated as the design cycle progressed based on the literature study findings.

4.2 Treatment design

The subsequent phase within the design cycle is called treatment design. Wieringa employs the term "treatment" instead of "solution" at this stage, as it remains uncertain whether the design will solve the problem. In this phase, the design of the IDS was finalized, comprising the most time-intensive aspect of the design cycle. The proposed design is covered in Chapter 5 and is one of this thesis' key contributions.

4.3 Treatment validation

The final phase of the design cycle is treatment validation. Here, the interaction between the artifact and its context is examined without actually implementing it within the real network. If the network serves as the context, validating the artifact requires network traffic that contains ground truth. Network traffic containing ground truth is challenging to acquire. This is because it is expensive to produce. NetFlow data, in particular, have some privacy constraints that increase complexity. Furthermore, it is crucial for the validation traffic to closely resemble real traffic in order to reflect how it would interact with the actual context.

Because of all these factors, the decision was made to generate network traffic containing ground truth data. The generation of ground truth data represents one of this thesis' contributions, and the details of this procedure are described in Section 6.2. Subsequently, the IDS was used on the ground truth data obtained from the two data sources. This yielded results on the performance of the IDS, which could assist in answering the research questions outlined in Section 1.3. The metrics defined in Section 2.3.1 were utilized to evaluate the performance.

Chapter 5
Proposed method

This chapter explains the proposed method to attain a more holistic view of a possible attack situation in the network. The objective is to leverage the potential of collaboration to achieve this holistic perspective. The method involves the development of a small IDS that integrates the outputs of 52 distinct anomaly detection methods and subsequently fuses the outputs. The fused alerts are then ranked on their perceived level of danger.

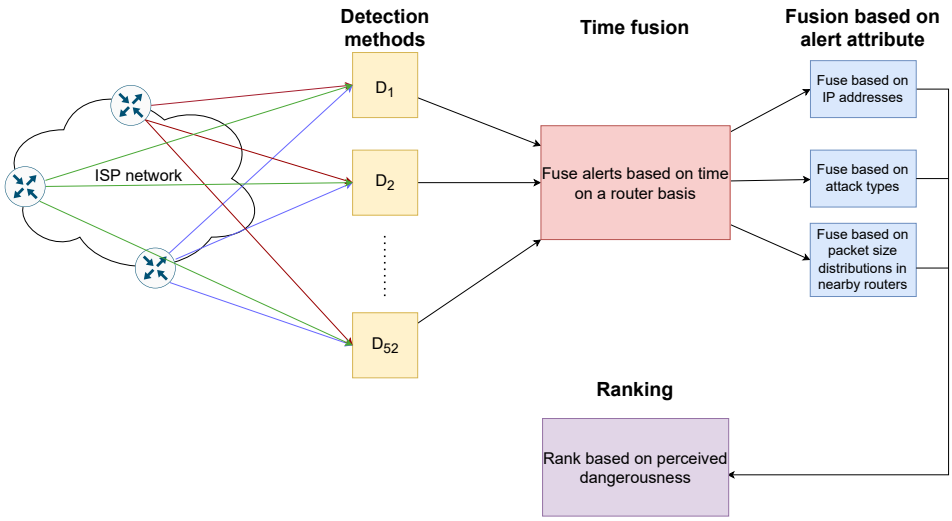


Figure 5.1: Architecture for the detection system

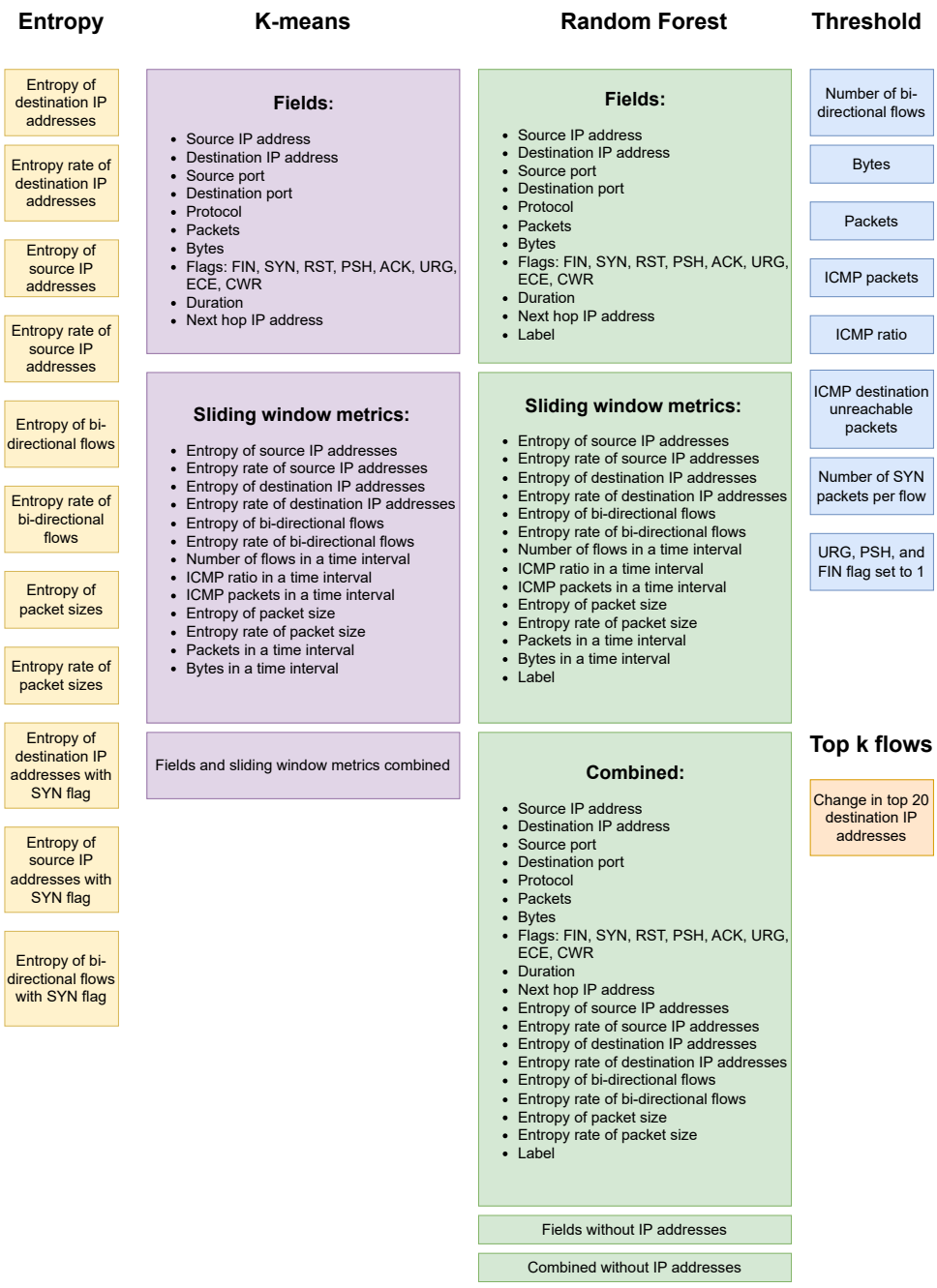


Figure 5.2: All detection methods on the NetFlow data

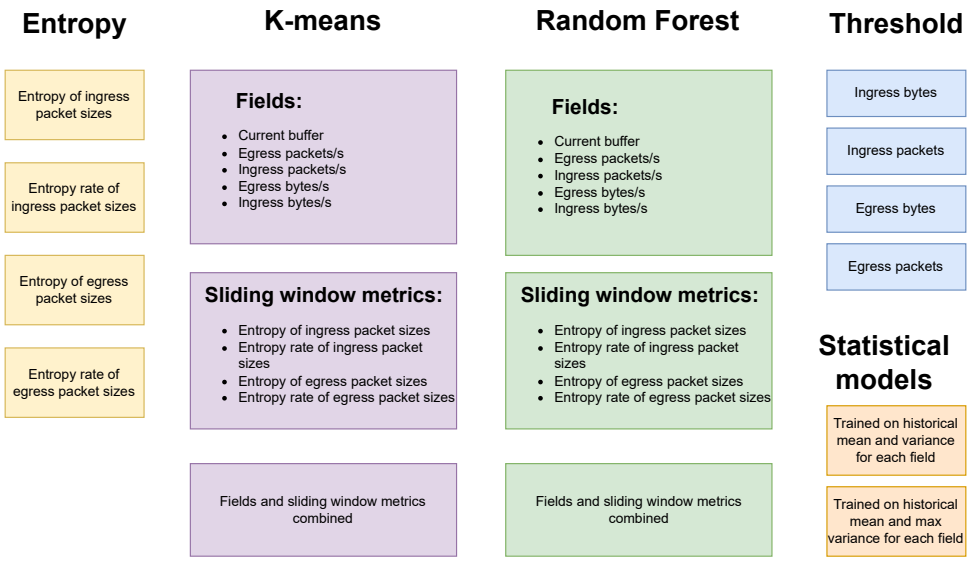


Figure 5.3: All detection methods on the Telemetry data

5.1 Alert fusion

The architecture of the IDS is illustrated in Figure 5.1, with the yellow boxes representing individual detection methods that transmit alerts to the time fusion unit. An overview of these 52 methods is presented in Figure 5.2 and Figure 5.3. This number is derived from the number of modules in the IDS that produce alerts. Each alert follows the format depicted in Listing 5.1.

Listing 5.1: Format of an alert from the detection methods

```
alert = {
    'sTime': possible start time of the event that caused
        the alert,
    'eTime': possible end time of the event that caused
       the alert,
    'Gateway': the router that caused the alert,
    'Deviation_score': normalized change from the mean of
        the comparison window,
    'Attack_type': possible type of attack,
    'IP': the IP address that caused the alert,
    'Packet_size_distribution': packet size distribution
       at the time of the event that caused the alert,
```

```
    'Weight': how likely the alert is to be a true
        positive based on historical data,
    'Real_label': whether the alert is really an attack
        or not,
}
```

In the time fusion unit, each alert is added to a collection of all alerts depending on what router it came from and when it was produced. Each minute this collection is scanned to find alerts overlapping in time to combine them. The combined alerts are sent to the ranking unit. If an alert includes a value in the "Attack_type" field, the alert from the detection method is also sent to the fusion based on attack types. Similarly, if the alert contains a value in the "IP" field, it is sent to the fusion based on IP addresses. If an alert contains a value in the "Packet_size_distribution" field, it is added to another collection of alerts that contain this field, depending on what router it came from and when it was produced. These alerts are combined based on the time and router and sent to the fusion based on packet size distribution. In the fusion units based on alert attributes, the alerts are combined further based on the fields before they are ranked based on perceived dangerousness.

5.1.1 Pre-processing and normalization

Each detection method represented by the yellow boxes in Figure 5.1 generates alerts based on their respective criteria. Except for machine learning methods and Xmas detection, all other methods generate alerts when the values surpass predefined thresholds. The selection of appropriate thresholds is crucial for optimizing the IDS's performance. To determine the suitable thresholds, each detection method is applied to a data set containing ground truth to measure which threshold performs best. The overall performance of the IDS is evaluated based on the metric used to define the best performance. In this case, TPR and F1 score were chosen as measures for the best performance. TPR was chosen to be maximized because if as many genuine alerts as possible are kept, then hopefully, the alert fusion will weed out the false alerts. The formula for calculating the TPR is defined in Equation (2.5). The decision to maximize the F1 score is motivated by assessing whether the IDS performs better when balancing the accuracy of alerts and the retention of true alerts. The selected thresholds derived from the maximum performance metric are subsequently applied to other data sets containing ground truth to evaluate the overall performance of the IDS.

Except for the machine learning methods and Xmas detection, all methods indicate how far the event that triggered the alert was from their definition of normal. This measure is referred to as the deviation score. Before transmitting the alerts, the deviation scores are normalized to facilitate comparison. Listing 5.1 illustrates

the format of an alert, and in this context, the deviation score is normalized using a variant of min-max normalization, ensuring that the score falls within the range of $[0, 1]$. The normalization process is outlined in Equation (5.1), where D represents the deviation score.

$$\text{Normalized } D = \frac{(\text{Actual } D) - (\text{Minimum Historical } D)}{3 \cdot (\text{Mean Historical } D) - (\text{Minimum Historical } D)} \tag{5.1}$$

The detection method is applied to historical data to determine the minimum and maximum values for normalization. Rather than employing the maximum value from the historical data, the maximum value is defined as $3 \cdot mean$ of the historical data. This approach is implemented to mitigate the potential impact of anomalies within the historical data that could distort the normalization process. However, it should be noted that not knowing the absolute maximum value may result in deviation scores exceeding the value of one.

The weight value in Listing 5.1 is utilized by the alert fusion component of the IDS to assess the likelihood of an attack occurrence. This value provides insights into the probability of an alert being a true positive. The weight calculation is based on the precision of alerts generated by the same detection method with identical parameters, employing a data set containing ground truth. Subsequently, these weights are applied when other data sets containing ground truth are used for performance evaluation. The precision of alerts can be found using Equation (2.4).

5.1.2 Time

Within the time fusion unit, the combination of alerts occurs based on whether they overlap in time. Each minute the alerts are combined into a summary alert based on if they occurred in that minute. If there are more alerts at this minute than a predetermined threshold, the summary alert is forwarded to the ranking unit. The volume of alerts is based on the weights in each alert, meaning that alerts with a lower likelihood of being a true positives contribute less to the overall volume of alerts on which the threshold is evaluated.

Furthermore, if an incoming alert contains the field “Packet_size_distribution”, it undergoes an additional algorithmic process. The pseudo-code for this algorithm is presented in Algorithm 5.1. This algorithm examines all alerts within the collection associated with the router that triggered the incoming alert and checks if these alerts overlap with the incoming alert based on time. When overlap is identified, the corresponding alert is merged with the overlapping alerts and transmitted to the unit responsible for further alert fusion based on packet size distribution.

The approach of this overlap in time is inspired by the fuzzy time methodology proposed in the research paper by Maggi, Matteucci, and Zanero [56]. When

comparing alerts with each other based on time, the start and end times of the alerts are evaluated by creating a fuzzy time interval using Equation (5.2). The concept of fuzzy time involves incorporating a predefined time interval at the beginning of the alert's time interval. This approach accounts for uncertainties in timestamping to ensure that no relevant alerts go unnoticed.

$$\text{Fuzzy time interval} = Interval(\text{start time - fuzzy time threshold, end time}) \quad (5.2)$$

5.1.3 Attack types

The first of the units that fuse the alerts based on alert attributes is the attack types. Each alert (except for the RF classifier alerts) contains a possible attack type. These assumed attack types are based on various research papers. "A novel feature-based framework enabling multi-type DDoS attacks detection" [21] proposed that one could determine if an attack was a LRDDoS attack or a flooding DDoS attack by looking at the entropy and entropy rate. They claimed that during a flooding DDoS attack, the number of different IP addresses in Equation (5.10) will increase substantially. This results in the entropy rate decreasing a considerable amount. However, they claim that during a LRDDoS attack, the number of different IP addresses is stable while there is a reduction in entropy. From this, it is concluded that if the entropy decreases and the entropy rate decreases, it is a LRDDoS attack. If only the entropy rate decreases, it is a flooding DDoS attack. This assumption holds for the destination IP address distribution, the source IP address distribution, and the bi-directional flow distribution.

Concerning the packet size distribution, a decrease in entropy indicates that the packet sizes in this interval are less random. This reduction in entropy could indicate that one packet size may be overly represented, which could lead to the assumption that the possible attack packets used the same protocol. If the entropy increases, it might indicate that the possible attack packets used different protocols.

For the K-means clustering, there can also be made some assumptions of attack type based on how the algorithm labels the clusters [61]. As Section 5.3.2 explains, if the Davies-Bouldin index approaches zero and one cluster is very compact, it will likely be because the same type of packet is prevalent in the possible attack cluster. These factors could indicate that the attack type is of the same protocol. If the Davies-Bouldin index is above a threshold and the attack cluster is less compact than the normal cluster, it may indicate that the possible attack uses different protocols.

To fuse the incoming alerts, the unit looks at the incoming alerts and groups them if they have the same attack type. It does this every minute and if the combined weight of the alerts in that minute is over a threshold, it will send a summary to the ranking unit.

5.1.4 IP addresses

The next unit fuses the alerts based on IP addresses in the alerts. A limited number of detection methods have access to the IP addresses. This is because, e.g., the telemetry data does not contain IP addresses, and the measurements using only the sliding window technique will alert based on an aggregation of flows that do not necessarily have the same IP address. This leaves the following detection methods:

- K-means clustering on flow fields
- K-means clustering on flow fields combined with sliding window measurements
- RF classifier on flow fields
- RF classifier on flow fields combined with sliding window measurements
- Threshold for the number of packets in an SYN flow
- Top K flows
- URG, PSH, and FIN flags set

This unit works similarly to the fusion on attack types but rather than looking at the attack type, it looks at the IP address in the alert. When it has accumulated enough alerts for an IP address in a minute, it will send a summary to the ranking unit.

5.1.5 Packet size distribution

A feature shared by telemetry and NetFlow data is the entropy of packet sizes. This entropy is computed by analyzing the probability distribution of various packet sizes within a specific time interval. Another insightful analysis that can be performed using this distribution is the calculation of Information Distance between two distributions. In the research paper titled “Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics” by Xiang et al. [34], Information Distance is employed to differentiate between benign and attack traffic. While drawing inspiration from their approach, the comparison scenario is modified. Instead of comparing benign and attack traffic, distributions obtained from different routers are compared. This comparison enables assessing if other routers exhibit similar packet size distributions. The compared routers can be considered “nearby” routers, where proximity is defined based on a predetermined threshold for the shortest path between the routers. If the unit identifies alerts from nearby routers with an Information Distance below a specified threshold within the fuzzy time interval, a summary of these alerts is forwarded to the ranking unit.

5.1.6 Ranking

The ranking unit receives the summaries of the alerts from the fusion units and displays them in a ranked manner. The ranking process is based on the perceived level of danger associated with each alert. Several factors contribute to this ranking, but the primary focus lies on attack types and deviation scores. Attack types are prioritized based on what is most dangerous to the network’s owner. In the case

of an ISP responsible for a core network, attacks compromising network resources such as bandwidth may be deemed the most severe. Consequently, various types of flooding DDoS attacks can be ranked the highest. The prioritization order of attack types in this thesis is as follows:

1. Flooding
2. ICMP Flood
3. SYN Flood
4. Same protocol
5. Xmas
6. LRDDoS
7. Different protocols

Other prioritization orders and their effect may be studied in future work.

Fused alerts are also ranked based on their deviation scores. Collecting the deviation scores from all alerts creates a list that includes all scores from the fused alerts. This list's mean and standard deviation are calculated in the ranking unit to compare the scores.

Upon the arrival of a combined alert in the ranking unit, it is added to a list of alerts scheduled for ranking. Every minute alerts older than 15 minutes are removed from the list, and the remaining alerts are sorted based on their perceived level of danger. Subsequently, the ranked alerts are written to a ranking file.

5.2 Statistical methods

5.2.1 Empirical mean-variance model

The first implemented method, inspired by [47], utilizes past network telemetry data to construct a statistical, empirical mean-variance model that represents normal network behavior. The model operates under the assumption that network traffic follows a weekly pattern. By examining Figure 5.4, it can be observed that the traffic pattern exhibits a similarity from week to week. To build the statistical model, n weeks of traffic data are gathered for a specific field on a particular router and stored in a data structure. This structure is organized based on the weekday, hour, and minute for each day of the week.

Once all the minutes for the n weeks have been added to the data structure, the mean for each minute is calculated. This mean is then denoised using FFT, which keeps the 50 highest energy coefficients. The data are denoised to remove any potential anomalies in the traffic data which would disturb the "normalcy" of the model. The difference between before and after denoising can be seen in Figure 5.5 and Figure 5.6. The denoised data are then stored in a data structure along with the

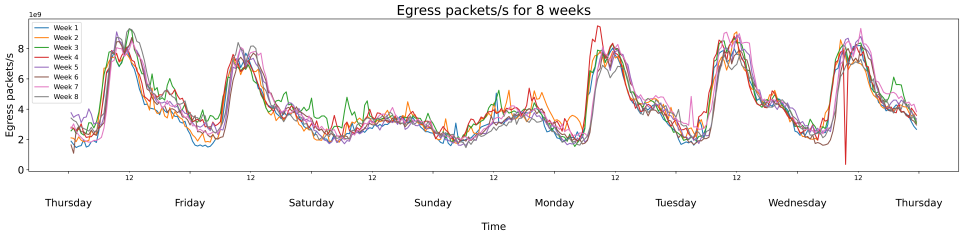


Figure 5.4: Packets/s for eight weeks

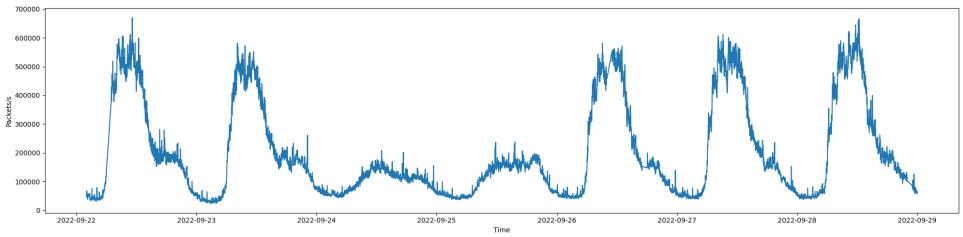


Figure 5.5: One week of data before denoising using FFT

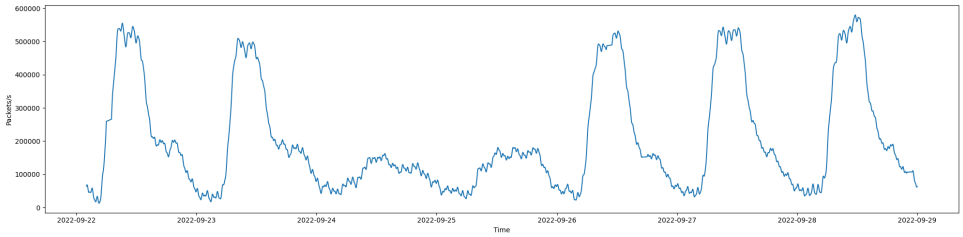


Figure 5.6: One week of data after denoising using FFT

standard deviation of the denoised mean obtained from the raw data. The structure's format is similar to that shown in Listing 5.2.

Listing 5.2: Data structure for mean and variance for the statistical model

```
{'weekday':
    {'0':
        {'hour':
            {'0':
                {'minute':
```

```
{'0': {'mean': 240072.505,
       'variance': 195377.556},
'1': {'mean': 235209.124,
      'variance': 197224.841},
'2': {'mean': 207929.144,
      'variance': 173719.152},
'3': {'mean': 202595.434,
      'variance': 182559.744},
. . .
```

The mean and variance derived from the statistical model serve as the foundation for detection. During detection, the current traffic for a given minute is compared with the corresponding minute of the corresponding weekday in the data structure. A deviation score is calculated using Equation (5.3), where $D(t)$ represents the deviation score, $T(t)$ denotes the current traffic for time t, $P(t)$ represents the mean for minute t, and $V(t)$ represents the variance for minute t.

$$D(t) = \frac{T(t) - P(t)}{V(t)} \tag{5.3}$$

The formula is derived from the equation for time series data (Equation (5.4)), where $A(t)$ signifies the anomaly component of the traffic.

$$T(t) = P(t) + V(t) + A(t) \tag{5.4}$$

When the deviation score $D(t)$ exceeds a predetermined threshold, the detection module generates an alert in the format shown in Listing 5.1. This statistical model is trained for five features:

1. Current egress queue size
2. Egress packets/s
3. Egress bytes/s
4. Ingess packets/s
5. Ingress bytes/s

Given that the weekly pattern is not flawless, the variance $V(t)$ can occasionally become significant. To address this, the deviation score using the maximum variance encountered for each specific feature during the training period is also calculated.

The relationship between the deviation score and the actual incoming traffic can be examined to get an idea of an appropriate threshold. From Figure 5.7, it can be observed that a deviation score of 20 corresponds to a probability of $1.78 \cdot 10^{-99}$, while in Figure 5.8, a deviation score of one yields a probability of $1.76 \cdot 10^{-9}$. These results suggest that a deviation score of 20 is approximately 10^{90} times less probable than a deviation score of 1. Such a substantial difference indicates a highly improbable event, concluding that the threshold should be below at least 20.

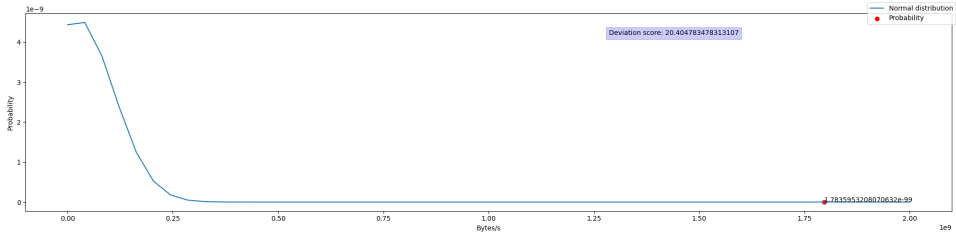


Figure 5.7: Deviation score of 20 compared to probability

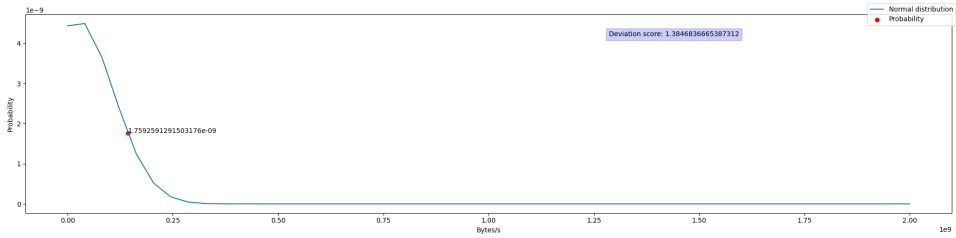


Figure 5.8: Deviation score of 1.38 compared to probability

Threshold

To detect possible attacks, different methods that base themselves on a predefined threshold were implemented. These predefined thresholds are determined by the best performance of the implemented method in detecting attacks on a set of ground truth data. The performance can be measured using the metrics presented in Section 2.3.1.

5.2.2 Threshold: packets and bytes

A simple threshold method is checking the number of packets and bytes in a specific time interval with a specific frequency. Zhou et al. [21] ranked the number of packets as the second most important feature for detecting DDoS attacks. The detection uses a sliding window method, visualized in Listing 5.3.

Listing 5.3: Sliding window functionality

```
values = [ 1, 2, 3, 4, 5, 6, 7, 8, 9]

value1 = [ 1, 2, 3, 4, 5]
value2 = [ 2, 3, 4, 5, 6]
value3 = [ 3, 4, 5, 6, 7]
value4 = [ 4, 5, 6, 7, 8]
value5 = [ 5, 6, 7, 8, 9]
```

In Listing 5.3, the sliding window technique is employed with a window size of five, sliding one element at a time. This approach can be applied to time series data by considering the window size as a time interval and sliding it with a specified time-frequency.

When counting the number of bytes and packets using a sliding window, the bytes and packets are aggregated within a designated window duration, such as 5 minutes. Then, with a specified frequency, e.g., 1 minute, the window is shifted forward by 1 minute, continuously summing the bytes and packets within the updated window. This methodology enables the observation of the evolution of the byte and packet counts over time and the detection of any sudden increases in traffic volume.

The current packet and byte count measurement can be compared to the previous $n \cdot frequency$ measurements, where n represents the comparison window. For instance, if the comparison window is set to 10 and the frequency is one minute, the current measurement would be compared with the measurements from the last 10 minutes. If the change surpasses a predefined threshold, the detection module generates an alert in the format illustrated in Listing 5.1.

5.2.3 Threshold: SYN flag

The threshold method described here is based on analyzing the TCP three-way handshake. As mentioned in the background (Section 2.1.2), the three-way handshake consists of an SYN packet, followed by an SYN-ACK packet, and finally, an ACK packet. In NetFlow, these three packets are counted as separate flows.

In the three-way handshake, the SYN packet is sent from the client to the server with the SYN flag set to 1. The subsequent SYN-ACK packet is sent from the server to the client, and since the IP addresses are reversed compared to the SYN packet, it is considered a different flow. Finally, the ACK packet is sent from the client to the server without the SYN flag set to 1, so it is not part of the first flow.

Under normal circumstances, a flow associated with the SYN flag set to 1 for the three-way handshake should only contain one packet. Additional packets may be present due to re-transmissions, but one packet is the standard. Therefore, a threshold is set such that any flows with the SYN flag set to 1 and a packet count exceeding a predetermined number will trigger an alarm. For instance, a SYN flow with 60 packets, as illustrated in Figure 5.9, might indicate a SYN flood attack.

5.2.4 Threshold: URG, PSH, and FIN flag

To try to detect Xmas attacks as explained in Section 2.1.2, a method that looks at the URG, PSH, and FIN flags in a flow is added. The detection method checks if a

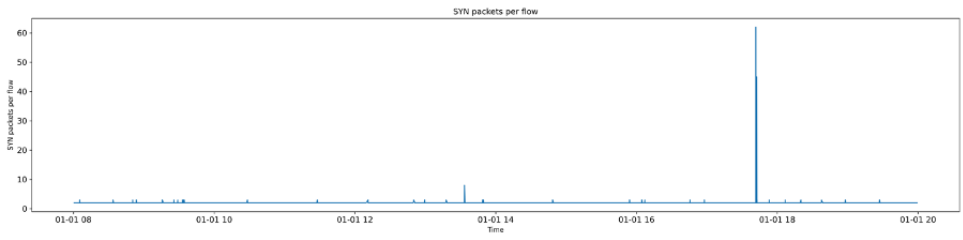


Figure 5.9: Number of packets in flows with the SYN flag set

flow has all these flags set to one simultaneously, which would be unusual [62]. If a flow meets these criteria, the detection method produces an alert.

5.2.5 Threshold: ICMP destination unreachable

Another threshold relies on the amount of ICMP destination unreachable packets within a specified time interval and frequency. It works like the sliding window for the packets and bytes threshold method. It will create an alert if the change from the last, e.g., 10 minutes, is over a predetermined threshold. A threshold for the number of ICMP destination unreachable packets is chosen because, during an attack, the destination host is overwhelmed by many attack packets and is thus unreachable for a request. Consequently, it will send out an ICMP destination unreachable packet to the originator of the request packet, and naturally, the count of ICMP destination unreachable packets will escalate during an attack [21]. The number of ICMP destination unreachable packets may also indicate a port scan since a closed port may send out this packet. A port scan can be a predecessor to an attack and is therefore helpful to notice [21].

5.2.6 Threshold: ICMP packets and ICMP ratio

There are also threshold methods based on the ICMP ratio and the number of ICMP packets. The decision to include these features was influenced by the research paper by Zhou et al. [21], which ranked them as the seventh and eighth most important, respectively. Thus, their incorporation into the detection system was deemed reasonable. Zhou et al. [21] claim that the main protocols utilized by DDoS botnets are ICMP and HTTP; therefore, it is interesting to look at these features. The ICMP ratio is calculated by the formula in Equation (5.5) in a specified time interval. The same sliding window method is used to detect anomalies in these measurements. As seen in Figure 5.10, the ICMP ratio during a 10-minute interval

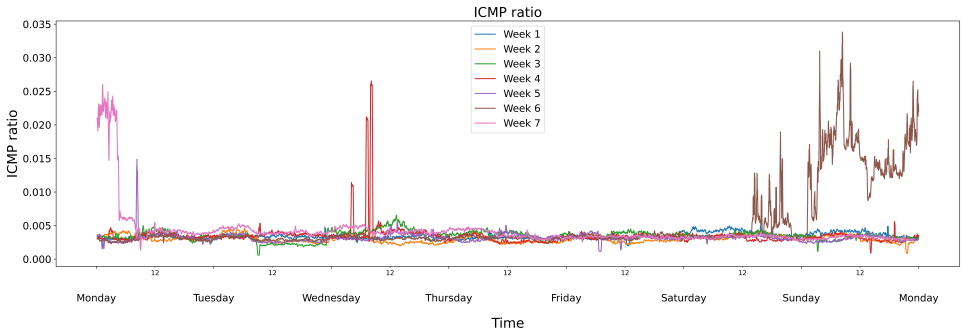


Figure 5.10: ICMP ratio in a 10-minute interval over a period of seven weeks

is pretty stable week to week and lies mostly below 0.005.

$$\text{ICMP Ratio} = \frac{\text{ICMP packets}}{\text{packets of all other protocols}} \tag{5.5}$$

5.2.7 Threshold: bi-directional flows

The last threshold method is on the number of bi-directional flows. A bi-directional flow can be defined as a flow where all packets go between two specific addresses. So a Flow $F = (IP_{src}, IP_{dst}) = (IP_{dst}, IP_{src})$. The packets for all these flows are aggregated in the NetFlow records using the sliding window method, and if the change precedes a predetermined threshold, the detection module will send out an alert. As seen in Figure 5.11, the number of flows is consistently below at least 300 000 in a 10-minute interval. This feature is selected because, during a DDoS attack, this is likely to increase, especially if the attack is a spoofing attack[1][34].

Entropy

Entropy calculations were also incorporated into the detection methods. These calculations were made based on the generalized entropy defined in Equation (2.8) with an α value of 10. This is the most extensive part of the detection system, with 15 different detection modules.

5.2.8 Entropy and entropy rate: packet size

For the telemetry data, two entropy metrics are calculated: the entropy of the packet size and the entropy rate of the packet size. These metrics are included because of

[1]The attacker changes its source IP address to trick the victim.

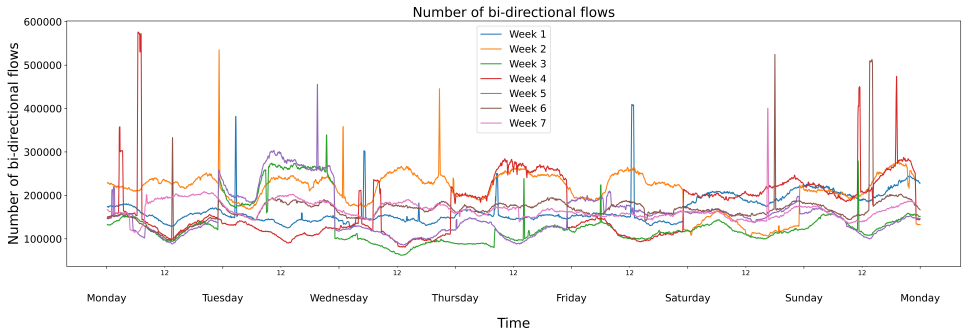


Figure 5.11: The number of bi-directional flows in a time interval of 10 minutes over a period of seven weeks

the findings in the paper [21], which concluded that the entropy of the packet size was the most important feature for detecting DDoS attacks, and the entropy rate of the packet size was the third most important feature.

The same sliding window technique described earlier is employed to calculate the entropy. This involves calculating the entropy at a predefined frequency, such as every 1 minute, over a predefined interval, such as the last 5 minutes of data. This means the entropy is calculated every minute based on the packet sizes observed over the last 5 minutes. If this entropy has significantly changed from the mean of a predefined comparison window, e.g., 10 minutes, there will be sent out an alert the same format as in Listing 5.1.

There is a need to determine the packet sizes from the available telemetry data to calculate the entropy. Since the telemetry data does not directly provide the packet size, bytes and packets per second measurements derived from the data are used. The packet size is estimated using the formula given in Equation (5.6).

$$\text{Packet size} = \frac{\text{bytes/s}}{\text{packets/s}} = \frac{\text{bytes}}{\text{packets}} \tag{5.6}$$

If the number of packets in the measurement is zero, the packet size will also be zero. The approach used by [21] is adopted to calculate the entropy of the packet sizes, which involves finding the probability distribution of the packet sizes. The probability distribution can be found by storing the number of occurrences of each specific packet size within the time interval in a data structure. Once all the packet sizes within the interval are accounted for, the total sum of packets is calculated. To obtain the probability distribution of the packet sizes in this time interval, each unique packet size is analyzed, and the number of packets of that size is divided by

the total sum of packets in the interval, as described in Equation (5.7).

$$P_{\text{packet size}}(i) = \frac{\text{Number of packets with packet } size_i}{\text{Total number of packets}} \tag{5.7}$$

This will be the probability of a particular packet size in this time interval. The entropy can then be found by using Equation (2.8), where $P(X)$ is the packet size distribution and α is the order of the entropy. The entropy rate is just the entropy divided by how many different packet sizes there are in the time interval, given by Equation (5.8).

$$\text{Entropy rate of packet sizes} = \frac{\text{Entropy of packet sizes}}{\text{Number of different packet sizes}} \tag{5.8}$$

The entropy rate calculated using the telemetry data may not be as accurate as with access to the actual packet sizes instead of just the bytes and packets per second. The average values over a second might not precisely reflect the actual packet sizes, although they can still offer some indication of the overall trend.

The packet size entropy and entropy rate can also be calculated for NetFlow data. This information can be derived from the number of packets and bytes in each flow. The packet size information from NetFlow data may be more accurate than the telemetry data since a flow is more likely to consist of similar packets, potentially resulting in a higher probability of being the same packet size.

5.2.9 Entropy and entropy rate: destination IP addresses

As per the findings presented in the research paper by Zhou et al. [21], the examination of Pearson correlation coefficients revealed a linear association between the entropy of destination IP addresses and the entropy of bi-directional flows. Consequently, the entropy of destination IP addresses was excluded from consideration. However, in this study, the decision has been made to include it for reinforcement purposes. Assuming they rank the same on the feature ranking, the entropy of destination IP addresses would hold the tenth position, while the entropy rate would rank fifth. The entropy of the destination IP address distribution tends to exhibit lower randomness during a DDoS attack, making it a valuable feature for detecting such attacks [63].

The computation of the destination IP address entropy involves storing the number of packets for each destination IP address within a specific time interval. The probability distribution is derived by determining the probability of occurrence for each IP address. Equation (5.9) defines this probability.

$$P_{\text{destination IP address}}(i) = \frac{\text{Number of packets with destination } IP_i}{\text{Total number of packets}} \tag{5.9}$$

The entropy rate can be obtained by dividing the entropy by the number of distinct destination IP addresses, as indicated in Equation (5.10).

$$\text{Entropy rate of destination IPs} = \frac{\text{Entropy of destination IPs}}{\text{Number of different destination IP addresses}} \tag{5.10}$$

5.2.10 Entropy and entropy rate: source IP address

According to [21], the entropy rate of source IP address was the fourth most important feature. This can be found the same way as the destination IP address entropy and entropy rate.

5.2.11 Entropy and entropy rate: bi-directional flow

As defined earlier a bi-directional flow is $F = (IP_{src}, IP_{dst}) = (IP_{dst}, IP_{src})$. The entropy of bi-directional flows can be obtained by first iterating through all NetFlow records. For each record, the source and destination addresses of the flow are examined, and the bi-directional flow is stored in a designated data structure. The number of packets associated with each bi-directional flow within the interval is recorded, enabling the probability calculation for each specific bi-directional flow. The probability is defined by Equation (5.11).

$$P_{\text{bi-directional flow}}(i) = \frac{\text{Number of packets for bi-directional } flow_i}{\text{Total number of packets}} \tag{5.11}$$

Subsequently, the entropy is computed based on the probability distribution of all flows within the interval. Furthermore, the entropy rate can be calculated by dividing the entropy by the total number of bi-directional flows, as outlined in Equation (5.12).

$$\text{Entropy rate of bi-directional flows} = \frac{\text{Entropy of bi-directional flows}}{\text{Number of different bi-directional flows}} \tag{5.12}$$

5.2.12 Entropy: SYN flows

In addition, entropy calculations were performed on just the flows with the SYN flag set. This inclusion aimed to introduce redundancy to the threshold established for SYN packets per flow. Consequently, the entropy values for both the destination and source IP addresses and the entropy of bi-directional flows were computed for the flows with the SYN flag set in the NetFlow records.

5.2.13 Top 20 destination IP address flows

The detection system's last detection method is based on a paper by Abdelkefi, Jiang, and Dimitropoulos [64]. They use something they call K-sparse Approximation to

compress the distribution of traffic volume over a specific feature. They can do this because they assume that the distribution exhibits a power-law decay when sorted from highest to lowest. In this thesis, it is used to detect if there is any change in the top 20 destination IP addresses based on the number of packets for each address. The detection module will send an alert with a deviation score calculated based on the change in position. Change in position is calculated by using Equation (5.13), where the last position is either 20 if the IP address was not present in the last top 20 flows the last minute or a lower position in the top 20 flows the last minute.

$$\text{Change in Position} = \frac{\text{Last Position} - \text{New Position}}{20} \tag{5.13}$$

The change in position will then be given by the percentage of positions the destination IP address flow "climbed" in the last minute. A change in position of 100% occurs if an IP address was not present in the last top flows at the last minute but is the number one top flow at the current minute.

5.3 Machine learning

The proposed IDS also includes two machine learning algorithms, namely RF and K-means. The RF classifier is an example of a supervised method, while K-means clustering is an unsupervised method. In implementing both a supervised and unsupervised learning algorithm, the different effects these types of algorithms have in detecting attacks can be investigated.

5.3.1 Random Forest

The RF algorithm is utilized as a component in the detection system. Numerous research papers in the literature have demonstrated promising outcomes in DDoS detection by implementing a RF classifier. Consequently, the RF algorithm was included as the only supervised learning approach. The model is trained using telemetry data, which comprises measurements taken at a 2-second frequency. The features utilized for training the RF model are illustrated in Figure 5.3.

The entropy and entropy rate of ingress and egress packet sizes were calculated within a specific time interval with a specific time-frequency, using the sliding window technique. The entropy values were calculated and subsequently incorporated into the respective row of each measurement recorded during the corresponding time-frequency. The labeling of each measurement is determined based on its timestamp by comparing it with a predefined list of timestamps indicating the presence of attack packets within the network.

The RF classifier is trained on this data set using 100 decision trees. The trained RF classifier is applied to the current data for anomaly detection. An alert is

generated if an anomaly is detected, containing the timestamp and the corresponding router information. During result analysis, the alert also includes the actual label of the data, facilitating the calculation of performance metrics for the IDS.

Additionally, the RF classifier was trained on flow records derived from NetFlow data, utilizing five sets of features. These sets include flow header fields, entropy calculations performed on the flows, and their combination. Figure 5.2 visualizes the different feature sets.

The reason for the number of bi-directional flows, ICMP ratio, number of ICMP packets, number of packets, and number of bytes not being included in the combination set is that they would be redundant with the header fields already there.

The classifier's training with and without IP addresses is driven by the limited IP address pool available for selection during attack generation. Training the classifier on data including IP addresses could lead to a bias where only these specific IP addresses are identified as abnormal, compromising accuracy. Thus, both scenarios were explored to assess the impact.

The labeling process involved matching the IP address in the flow record against a list of known attacker IP addresses, the known victim IP address, and attack timestamps. The combined datasets comprised flow records captured over a specific period. The entropy features are calculated within an interval for a specific frequency and added to the row of every flow within this time-frequency. The RF classifier was trained on the different data sets, employing 100 decision trees as with the telemetry data.

5.3.2 K-means

To incorporate additional valuable information, the unsupervised clustering algorithm K-means is utilized. To identify the attack cluster, the approach outlined in the research paper by Petrovic et al. [61] is employed. This approach combines the Davies-Bouldin index and a comparison of centroid diameters between clusters. The rationale behind this method is that during a massive attack, the attack cluster tends to be the largest yet highly compact[2]. Therefore, during a massive attack, the Davies-Bouldin index approaches zero, as does the cluster centroid diameter. In contrast, during normal traffic, the normal cluster should exhibit greater compactness than the attack cluster.

The K-means algorithm is applied to three different feature combinations for the telemetry data. The first combination solely consists of the field values explained in Section 5.3.1. The second combination includes the entropy and entropy rate of

[2]indicating similar traffic patterns

the packet sizes. Finally, the third combination combines both sets of features. The K-means algorithm clusters the data into two clusters, and the clusters are labeled using the algorithm described in [61]. The detection module generates an alert if a data point is assigned to the attack cluster. The Davies-Bouldin index and centroid diameters are utilized to determine the presence of a massive attack in the traffic, which is included in the alert as the attack type, as shown in Listing 5.1.

Regarding the NetFlow data, the K-means algorithm is applied to three different feature combinations. The first clustering is performed solely on the fields as depicted in Figure 5.2. The second clustering is conducted exclusively on the sliding window measurements. Lastly, the algorithm is applied to a combination of all these features. Like the telemetry data, an alert is generated if a feature combination is assigned to the attack cluster determined by the algorithm presented in [61].

Algorithm 5.1 Pseudo code for aggregation on packet size distribution

```
1: function AggregateAlerts(stime, etime, gateway, alert)
2:     fuzzyStartTime ← stime − 1 minute
3:     interval ← Create interval(fuzzyStartTime, etime)
4:     overlappingAlerts ← 0
5:     deviation_scores ← []
6:     real_labels ← []
7:     attack_types ← {}
8:     distributions ← []
9:     removeTimes ← []
10:    add alert to collection(gateway, interval, alert)
11:
12:    for each time in time intervals for this router do
13:        if start of time < stime − 15 minutes then
14:            append(removeTimes, time)
15:            continue
16:        end if
17:        if interval overlaps with time then
18:            alerts ← getAlerts(gateway, time, alertCollection)
19:            for each alert in alerts do
20:                append(deviation_scores, alert.Deviation_score)
21:                append(real_labels, alert.Real_label)
22:                attack_types[alert.Attack_type] += alert.Weight
23:                append(distributions, alert.Packet_size_distribution)
24:                overlappingAlerts += alert.Weight
25:            end for
26:        end if
27:    end for
28:
29:    for time in removeTimes do
30:        remove time interval from collection(gateway, time)
31:    end for
32:    if overlappingAlerts > 0 then
33:        message ← {"sTime" : stime,
34:                   "eTime" : etime,
35:                   "Gateway" : gateway,
36:                   "Deviation_scores" : deviation_scores,
37:                   "Real_labels" : real_labels,
38:                   "Attack_types" : attack_types,
39:                   "Packet_size_distributions" : distributions,
40:                   "alert_collection" : alertCollection,
41:                   "Weight" : overlappingAlerts}
42:        send message to packet size fusion
43:    end if
44: end function
```

Chapter 6
Results and discussion

6.1 Implementation and setup

Real attack traffic was needed to properly validate the results of the IDS. This attack data was obtained by generating four attacks within an ISP network. The attacks were generated using a custom-developed docker container [65] deployed on various servers within the ISP network. These docker containers contained the necessary attack scripts and logging capabilities to monitor the attack process. The cron job scheduler [66] was utilized to start the attack simultaneously across all docker containers. The attack procedure was controlled by a unified script containing the attack commands, attack durations, and breaks. This made the machines pose as a small botnet.

It was necessary to know which routers interacted with the attack traffic to retrieve the ground truth from the network. This information was gathered by executing the traceroute command before and after each attack to determine the path of the attack traffic. Additionally, the traceroute provided information regarding the availability of the victim server at specific times. Curl commands were also executed before and after each attack to verify the availability of the victim server. Through traceroutes, it was determined that 16 routers were involved in the first three attacks. After examining the NetFlow configuration of these routers, it was observed that only six routers had flow monitoring enabled on the interfaces traversed by the attacks. This configuration was likely due to the general practice of monitoring flows exclusively on customer interfaces, excluding core routers along the path. However, there was one exception among the 16 routers.

Consequently, ground truth data generated during the attacks was not captured by all routers in the path. Nonetheless, telemetry data from router interfaces were collected on nearly all interfaces, providing ground truth data for all 16 routers. As a result, the 28 detection methods utilizing NetFlow data were executed on the six routers with available ground truth. In comparison, the 16 detection methods

utilizing router interface telemetry data were executed on all 16 routers.

As mentioned in Section 2.4, the difference between the NetFlow data and telemetry data is that the NetFlow data set is the aggregation of packets into flows. In contrast, the telemetry data set contains the interface-based measurements of the traffic load of a router.

The latest version of the Apache HTTP web server was utilized for the first two attacks. However, for the third and fourth attacks, the web server was downgraded to version 2.2.34 to make it vulnerable to SlowLoris, Slow Read, and R.U.D.Y attacks. This downgrade was justifiable, as explained in Section 2.1.2, considering the substantial number of web servers still employing this version or earlier.

In the first three attacks, there were four attackers and one victim, all located within the ISP network. However, for the fourth attack, access was granted to four cloud computers located outside the ISP network. The technical specifications of the machines employed to generate the ground truth data are presented in Table 6.1. Once the attacks were generated, the data sets containing the ground truth data were

	Machine type	**CPUs**	**Memory**	**NIC speed**
ISP attackers	Lenovo ThinkServer RD350	16	32 GB	10 Gbits/s
Cloud attackers	Amazon VM	2	8 GB	Elastic
Victim	Dell Inc. PowerEdge R620	16	32 GB	1 Gbits/s

Table 6.1: Technical specifications of machines used in ground truth generation

obtained from the ISP Sikt, responsible for the Norwegian NREN. The machines used to generate the attacks were also provided by Sikt.

The IDS was developed using Python version 3.10.6. Pandas data frames [67] and NumPy arrays [68] were employed to handle the raw data in a standardized format. The machine learning algorithms were implemented using the Scikit-learn Python module [69]. The MQTT protocol based on a publish-subscribe model was utilized for communication between units within the IDS. The open-source Eclipse Mosquitto broker implemented the MQTT communication [70]. The shortest path between routers was determined using the NetworkX Python library [71], which enabled the construction of a graph representing the relevant section of the Norwegian NREN affected by the attacks. The complete implementation of the IDS can be accessed on GitHub[1].

[1] https://github.com/linneagustavsen/master-thesis

6.2 Ground truth generation

It was necessary to incorporate known attacks into the network data used for testing the methods to evaluate the performance of the detection methods. These attacks were incorporated into the data by generating attack data using various DDoS attack tools. This process was repeated with different variations four times. The source and destination IP addresses of these known attacks were then utilized to label the flows, train the RF classifiers, and assess the performance of the IDS.

An overview of the network topology can be seen in Figure 6.1, where the red arrows indicate the locations where NetFlow records are sampled. The flows are sampled with the rate: 1 out of 100 packets. Each router is identified by an abbreviation, with "AR" representing attack routers, "CR" representing core routers, and "VR" representing the victim router. There are more connections between routers that are not in the figure, but they have been omitted to simplify the representation.

6.2.1 Attack generation: round 1

The initial round of attack generation consisted of four distinct attack types: TCP SYN flood, SlowLoris, ping flood, and R.U.D.Y. The schedule for the attack procedure is presented in Table 6.2, with breaks between the attacks to differentiate between them. A tool called Hping3 [28] was used for generating the flooding attacks, while

Attack	**Duration [min]**
SYN Flood	5
Break	15
SlowLoris	13
Break	7
Ping Flood	7
Break	20
R.U.D.Y	15

Table 6.2: First attack generation procedure

SlowLoris and R.U.D.Y attacks were generated using the Slowhttptest tool [29].

6.2.2 Attack generation: round 2

The subsequent round had the same attack types but with slightly adjusted durations. The TCP SYN flood and the breaks between attacks were extended. Quite a similar round of attacks was generated because the SlowLoris attack stopped by itself due to

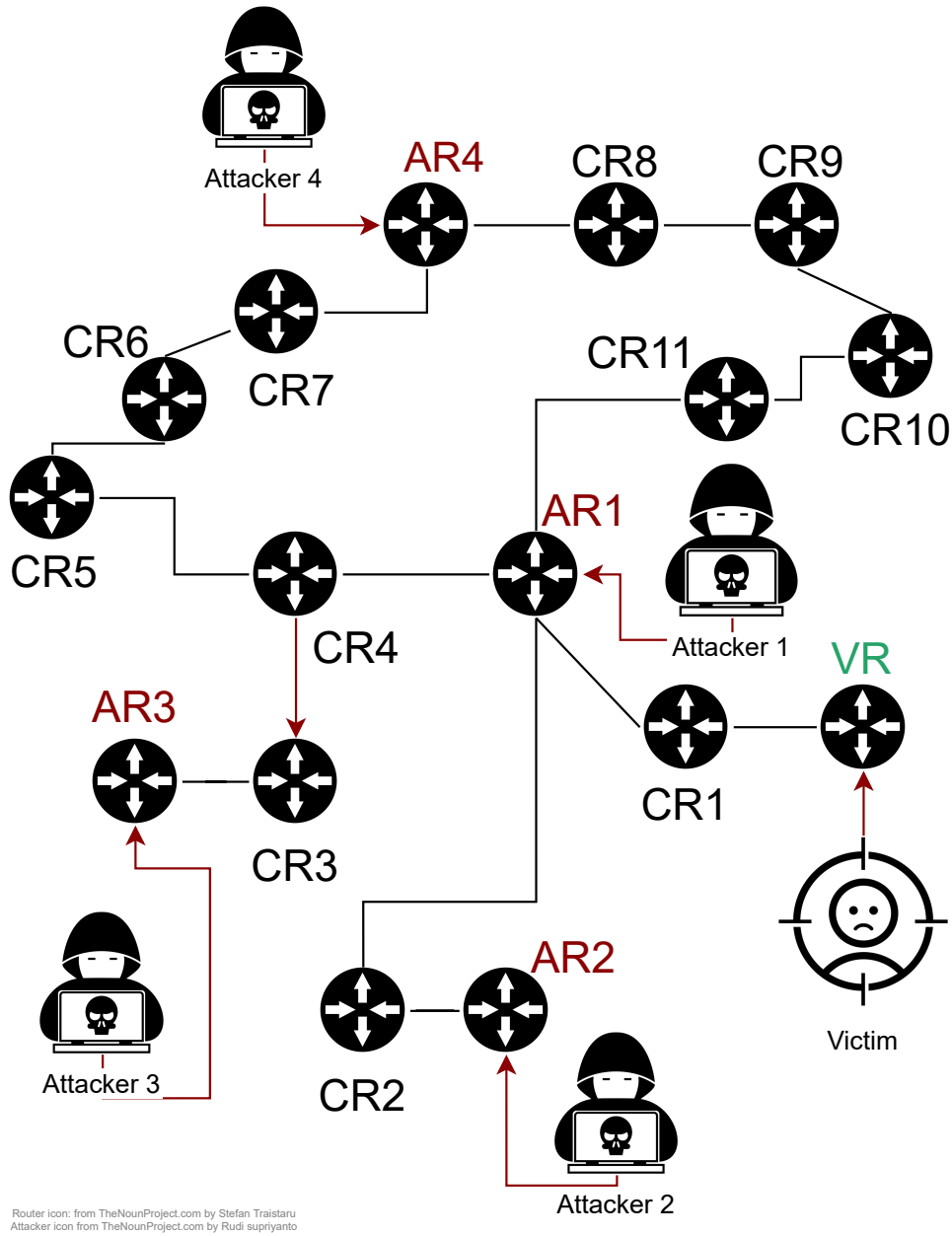


Figure 6.1: Topology over the network used in the ground truth generation

Attack	**Duration [min]**
SYN Flood	7
Break	30
SlowLoris	13
Break	7
Ping Flood	7
Break	40
R.U.D.Y	15

Table 6.3: Second attack generation procedure

the server closing the connections to protect itself. This closure was because it was not vulnerable to the SlowLoris attack, and it happened after only 3 minutes, which resulted in minimal attack traffic from the SlowLoris attack in the first iteration. Therefore, in this attack generation, the attack was restarted if it stopped within the attack duration. Additionally, the break between the SYN flood and SlowLoris attack was lengthened to ensure sufficient recovery time for the server before initiating a new attack.

6.2.3 Attack generation: round 3

The third round was more extensive than the preceding two, incorporating additional attack types and combinations of attacks. The attack schedule is outlined in Table 6.4.

Multiple attack scenarios were generated to be used for, among other things, training and testing the RF classifier. Using different data sets can avoid potential overfitting that could occur if the classifier were trained solely on the same attack types. Combining attacks was also relevant because attackers sometimes conceal LRDDoS attacks like SlowLoris behind more prominent flooding attacks, diverting attention and allowing the LRDDoS attack to continue unnoticed.

6.2.4 Attack generation: round 4

The final attack generation round was generated slightly later than the first three, as access to four additional machines was obtained, enabling a larger-scale attack. The attack procedure was identical to the third round, adding a form in the web server specifically for the R.U.D.Y attack, providing a target for utilizing the HTTP POST method. Given the availability of four additional machines, a more substantial attack was generated, which could prove valuable for detection purposes. The attack could also introduce valuable insights into detecting attacks originating from outside the

Attack	**Duration [min]**
UDP Flood	3
Break	10
SlowLoris	13
Break	20
Ping Flood	5
Break	7
Slow Read	13
Break	14
Blacknurse	10
Break	3
SYN Flood	7
Break	20
R.U.D.Y	9
Break	6
Xmas	4
Break	3
UDP Flood and SlowLoris	15
Break	7
Ping Flood and R.U.D.Y	10
Break	20
All attack types	12

Table 6.4: Third and fourth attack generation procedures

ISP network. The attack generation round followed the same procedure as round three, as shown in Table 6.4.

6.3 Detection methods

This section aims to analyze the outputs generated by each detection method to address research question **R1**, as defined in Section 1.3. The effect of the detection methods will be explored by examining the outputs produced by different anomaly detection methods when applied to real traffic data containing actual attacks.

This section will present plots illustrating the outputs of the methods. Most of these outputs will be from the third round of the attack generation, as this round had outputs from all attack types. This round is also primarily used to limit the volume of results presented compared to if results from all rounds would be present. The

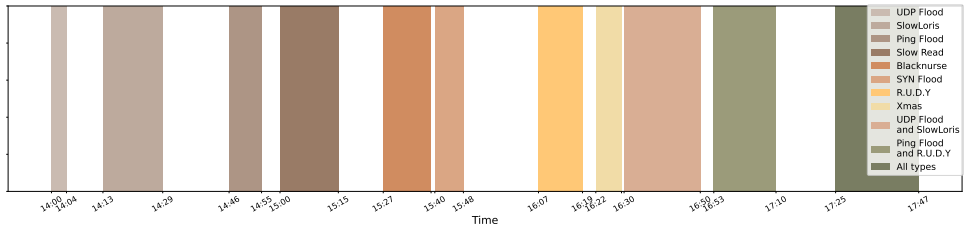


Figure 6.2: Example of background shadings indicating periods of attack during the third round of attack generation

plots from the outputs will include shaded areas indicating the periods during which attacks occurred, along with information about the specific attack types. An example of this shading is presented in Figure 6.2. The x-axis in the plots will represent the time of day when the outputs were generated, while the y-axis will vary depending on the chosen method.

6.3.1 Empirical mean-variance model

The empirical mean-variance model was trained using five features for all 16 routers with known ground truth. These models provided ten different outputs for each router. Specifically, the deviation scores were obtained for the five features based on the historical variance for the particular minute and the maximum historical variance for that feature.

Current egress queue size The first feature considered was the current egress queue size. Figure 6.3 shows the deviation scores during attack generation round 3 for the first hop router from the victim, VR, presented on a logarithmic scale. Furthermore, by examining Figure 6.4, there seems to be an indication of the router, CR5, being impacted by the attack, even four hops away from the victim and three hops away from an attacker. This observation suggests the possibility of detecting the attack earlier within the network before it reaches the victim. For an ISP, this could imply the ability to protect their customers from attacks before the attack packets even reach the customer network.

Egress packets/s The second feature considered in the empirical mean-variance model was the number of packets per second exiting the routers. Figure 6.5 displays the deviation scores for a router CR11, located four hops away from the victim and five hops away from an attacker. In this case, the deviation scores remain somewhat elevated throughout the entire duration. However, there appears to be some correlation between the periods of flooding attacks and an increase in the

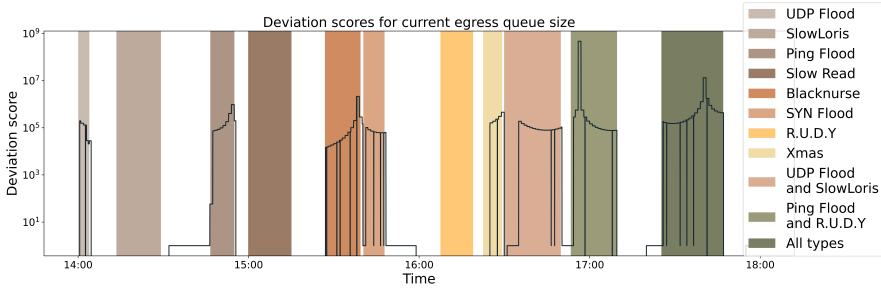


Figure 6.3: Deviation scores for queue size during attack generation round 3 on VR on a logarithmic scale

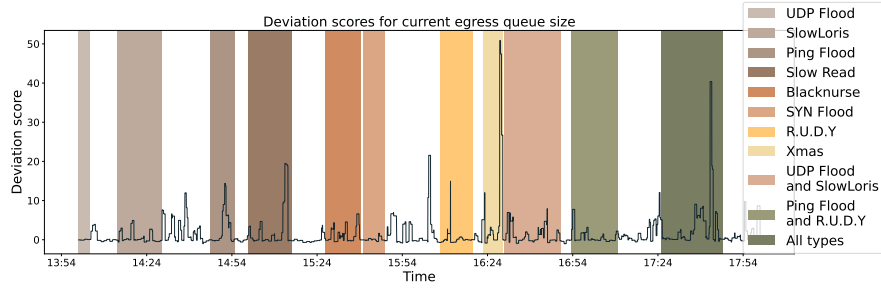


Figure 6.4: Deviation scores for queue size during attack generation round 3 on CR5

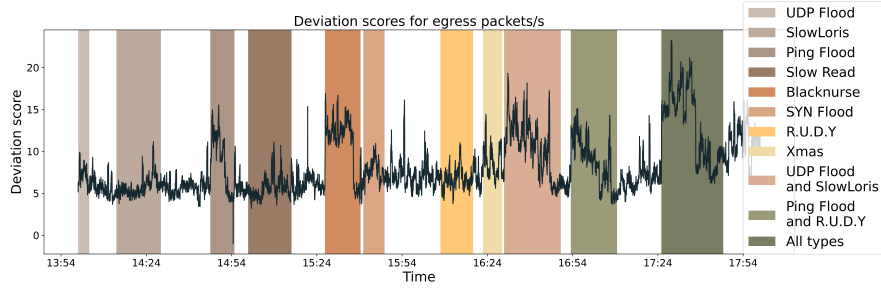


Figure 6.5: Deviation scores for egress packets/s during attack generation round 3 on CR11

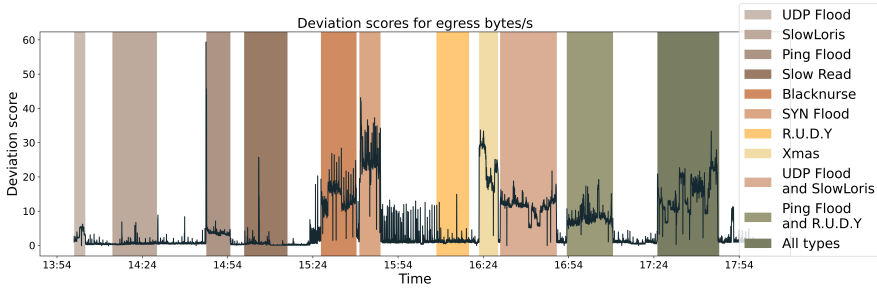


Figure 6.6: Deviation scores for egress bytes/s during attack generation round 3 on VR

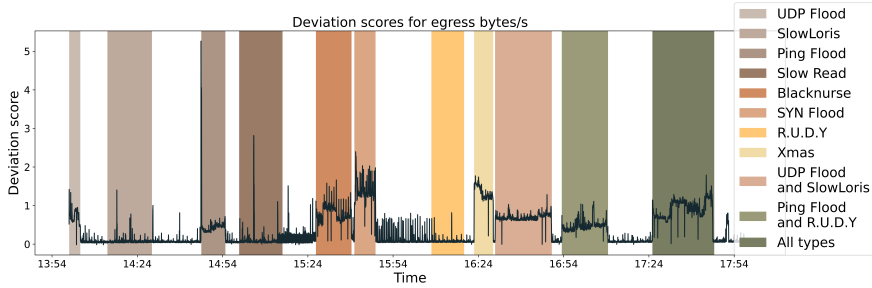


Figure 6.7: Deviation scores for egress bytes/s during attack generation round 3 on VR using the maximum variance

deviation scores. One possible explanation for the consistently high deviation scores could be that the training period was insufficient to establish a definitive pattern for this particular router. Another contributing factor could be a shift in the routed traffic passing through this gateway compared to the training period. This underscores the significance of having a sufficiently long training period that is regularly updated to reflect the current network conditions and exhibit stable behavior.

Egress bytes/s The last feature considered for outgoing traffic is bytes per second. Figure 6.6 displays the deviation scores for the router VR, while Figure 6.7 shows the same but using the maximum variance instead of the actual variance. These two figures are similar except for their difference in scale, but they have some slight differences. One noticeable distinction occurs during the SlowLoris attack. Figure 6.7 shows a slight increase in the deviation score shortly after the attack starts. However, in Figure 6.6, the deviation score does not exhibit the same proportional increase. Contrarily, during the R.U.D.Y attack, there is a slight increase in the deviation

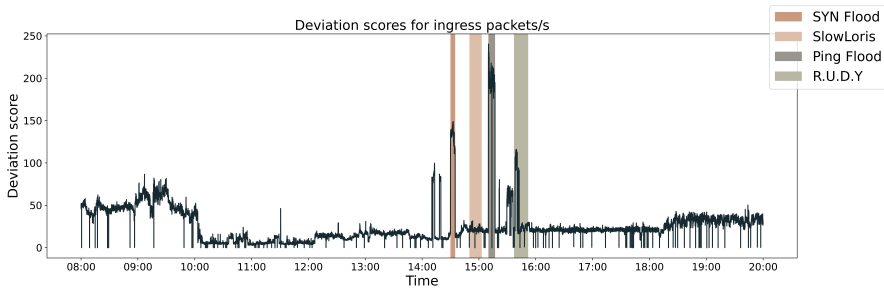


Figure 6.8: Deviation scores for ingress packets/s during attack generation round 1 on VR

score that is proportionally higher in Figure 6.6 compared to Figure 6.7. Additionally, the detection method utilizing the maximum variance demonstrates an increase in the deviation score after the Slow Read attack, which is not apparent in the other detection method. This difference might be attributed to a regular packet increase during that particular time of day, as it is not observed in the method employing the historical variance.

Ingress packets/s Turning the attention to the incoming packets per second, Figure 6.8 reveals a notable increase in deviation scores during the attack period compared to the rest of the day. In the case of this router, VR, the ingress packets per second represent the feedback packets that the victim sends in response to the attack packets. Interestingly, even during the onset of the R.U.D.Y attack, which is a LRDDoS attack, there is an observable increase in the deviation score. Furthermore, there are instances of elevated scores during the morning hours, which could potentially be attributed to unidentified attacks or special events resulting in increased packet activity.

Ingress bytes/s Lastly, detection of DDoS attacks was attempted using the deviation scores derived from the incoming bytes per second. By referring to Figure 6.9, it becomes clear that these scores represent the feedback traffic from the attacks, as there is no distinct indication of an attack during the attack period. From a logical standpoint, it is generally easier to identify attacks through the deviation scores of packets per second rather than bytes per second, as feedback packets tend to be smaller in size. Since the victim receives a substantial amount of attack traffic, it may attempt to send numerous small feedback packets, which could explain the observed increase in packets per second during the attacks. Additionally, in Figure 6.9, a similar increase in deviation scores can be observed during the morning hours. This

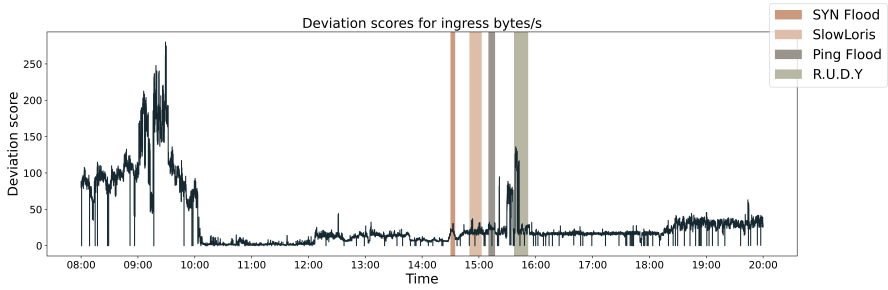


Figure 6.9: Deviation scores for ingress octets/s during attack generation round 1 on VR

increase is more prominent than the packets per second, suggesting the possibility of a significant influx of large packets sent into the router during that time.

6.3.2 Threshold: packets and bytes

Furthermore, the detection of DDoS attacks was explored based on analyzing packet and byte counts transmitted through the routers. Figure 6.10 provides a visual representation of the number of bytes within a five-minute window for both the router interface telemetry and NetFlow data. It is important to note that the output in this figure is collected from router VR, with the NetFlow data representing, among other things, the feedback traffic from the victim. In contrast, the router interface telemetry data set is collected from all interfaces, which means the figure shows the ingress traffic across all interfaces on that router. Consequently, the bytes from the telemetry data set are typically around 100 times larger than the NetFlow data, because they encompass traffic from all interfaces or because NetFlow samples every 100th packet. This distinction also explains why specific increases in bytes during the LRDDoS attacks may not be as pronounced in the telemetry data set as they are combined with traffic from many other interfaces. Regarding packet counts, Figure 6.11 illustrates that the patterns are more evident in the NetFlow data while less apparent in the telemetry data. However, slight increases can still be observed during flooding attacks. The difference in scale displayed in Figure 6.11 is most likely because the telemetry data set is collected from all interfaces, not solely customer interfaces.

6.3.3 Threshold: SYN flag

To investigate the impact of SYN flood attacks, a hypothesis was formulated suggesting that the number of packets within a flow with the SYN flag set to one would increase during such an attack. To validate this hypothesis, a threshold was

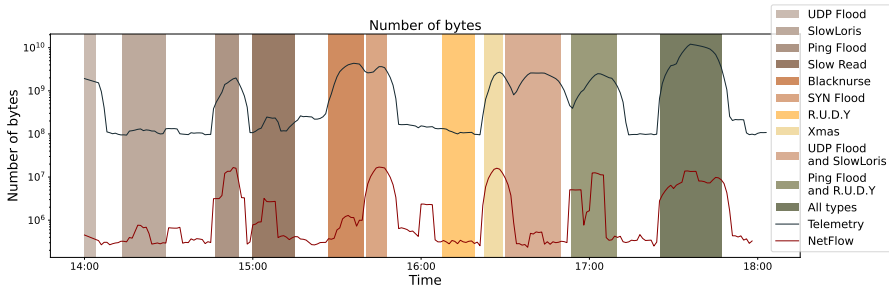


Figure 6.10: The number of bytes for attack generation round 3 on VR

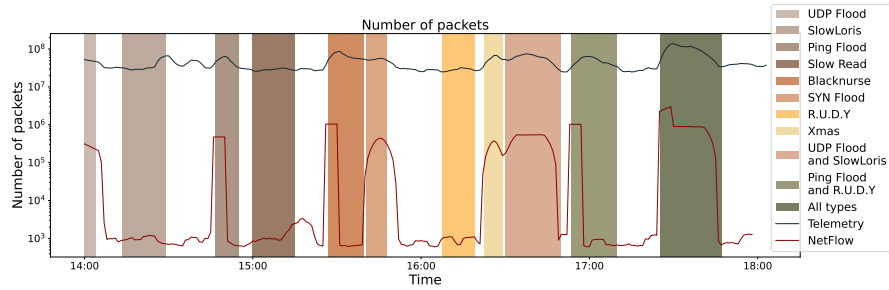


Figure 6.11: The number of packets for attack generation round 3 on AR4

implemented. Figure 6.12 displays the data, indicating that the hypothesis may hold, as there are flows with up to nine packets during the SYN flood attack.

However, when examining Figure 6.13, it becomes apparent that the accuracy of the hypothesis is not as consistent. Many other flows also exhibit a high number of packets, which challenges the idea that such flows are associated with the generated attacks. Notably, some of the most significant flows detected did not originate from the generated attacks, further weakening the accuracy of the hypothesis.

6.3.4 Threshold: URG, PSH, and FIN flag

Since a Xmas attack was launched in the ground truth generation, it would be interesting to see if it could be detected using a check on an unusual combination of TCP flags. Notably, this combination was only detected by the routers closest to the attackers. The one core router with NetFlow monitoring configured did not record these flows. The absence of these flows in the NetFlow records can be attributed to the fact that the sampling for NetFlow is performed on incoming traffic from CR4.

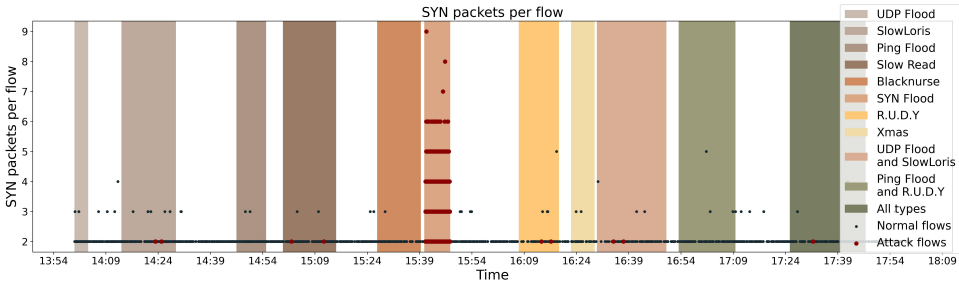


Figure 6.12: The number of packets with the SYN flag set for attack generation round 3 during the period of the SYN flood attack on AR1

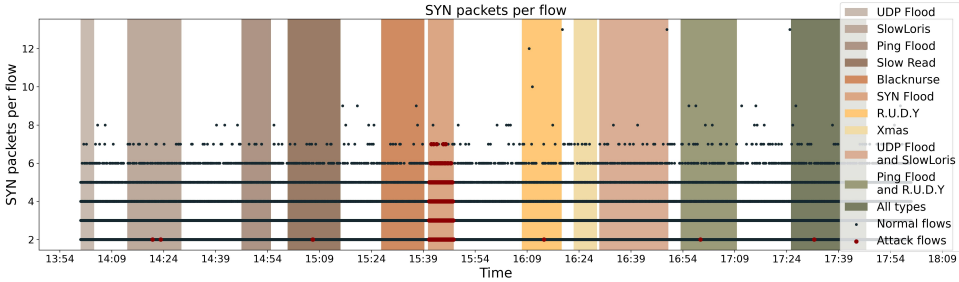


Figure 6.13: The number of packets with the SYN flag set for attack generation round 3 during the period of the SYN flood attack on AR2

The traffic from CR4 to CR3 is the feedback traffic from the victim which would not contain the Xmas attack.

Figure 6.14 showcases the flows triggered by this check. Interestingly, not solely the flows sent during the generated Xmas attack triggered this detection. The router CR2, responsible for detecting these flows, is connected to an Internet Exchange Point (IXP), a point of entry for traffic originating from other ISPs. The additional flows observed in Figure 6.14 could potentially represent another attempted attack originating from the network of a different ISP. Upon examining the destination ports of the flows between approximately 14:05 and 14:15, it becomes apparent that they are all destined for the same port but originate from different source ports, which could strengthen the hypothesis.

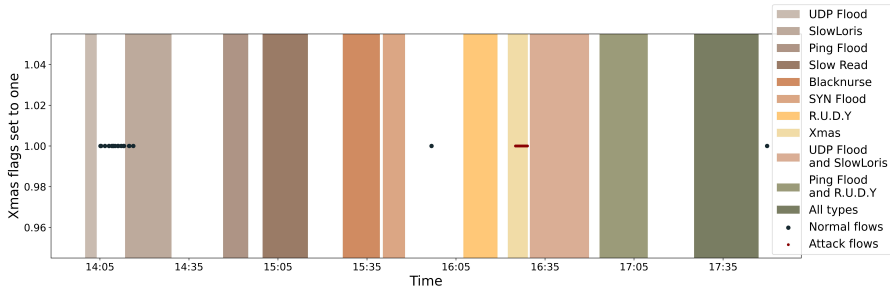


Figure 6.14: Flows with the URG, PSH, and FIN flags set for attack generation round 3 on AR2

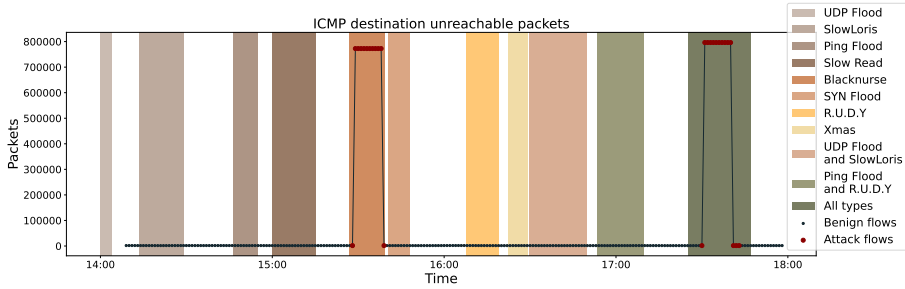


Figure 6.15: The number of ICMP destination unreachable packets for attack generation round 3 on AR2

6.3.5 Threshold: ICMP destination unreachable

The amount of ICMP destination unreachable packets within a 10-minute window can be observed in Figure 6.15. Notably, the figure reveals a noticeable increase in the number of packets during the Blacknurse attack and the combination of all attacks. This finding aligns with expectations, considering the attacker floods the victim with ICMP destination unreachable packets. Consequently, this feature enables source-side detection[2] of the Blacknurse attack.

In the study by Zhou et al. [21], the researchers suggested that this feature would be suitable for detecting attacks since the victim tends to send ICMP destination unreachable packets when overwhelmed with requests. Turning the attention to Figure 6.16, it can be examined whether the number of ICMP destination unreachable packets the victim sends back to the attackers is noticeable. It appears that there

[2]detecting attacks at the source of the attack packets

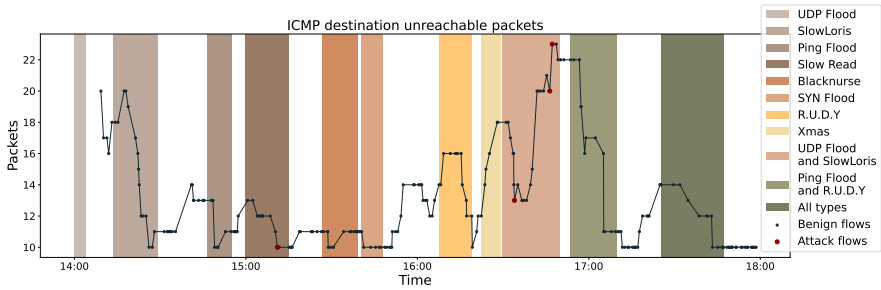


Figure 6.16: The number of ICMP destination unreachable packets for attack generation round 3 on VR

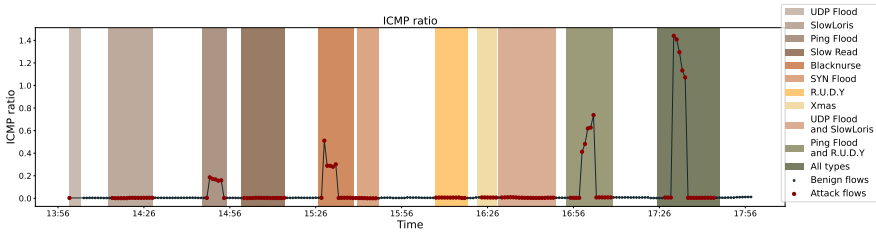


Figure 6.17: The ratio of ICMP packets to other packets coming into AR1

is a slight increase at the onset of the SlowLoris attack, possibly indicating the unavailability of the service but it is not caused by the generated attacks. However, this increase does not appear to be uncommon, given the considerable variation in the number of packets. The highest number of packets can be observed towards the end of the UDP flood and SlowLoris attack, potentially indicating that the attackers achieved their objective of rendering the service unavailable.

6.3.6 Threshold: ICMP packets and ICMP ratio

Analyzing the number of ICMP packets and the ICMP ratio provides valuable insights into the presence of increased ICMP traffic. This is evident in Figure 6.17 and Figure 6.18. These figures exhibit distinct increases during all attacks involving ICMP packets, as indicated by both metrics. Notably, no other periods display a comparable level of increase. Based on this clear distinction, one can speculate that these metrics hold promise in detecting ICMP-related attacks.

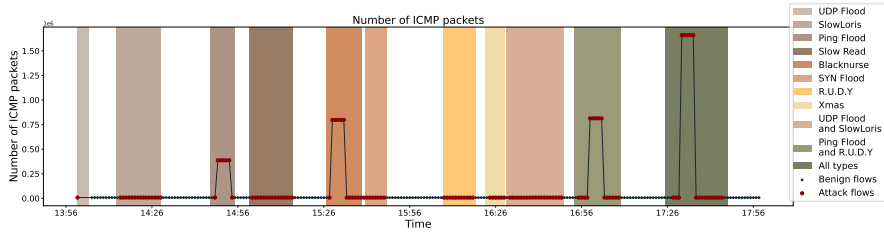


Figure 6.18: The number of ICMP packets coming into AR1

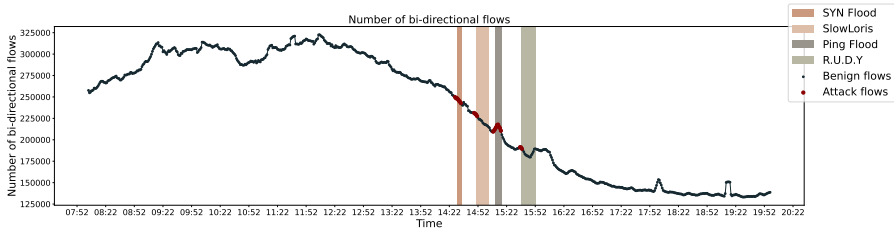


Figure 6.19: The number of bi-directional flows for attack generation round 1 on AR2

6.3.7 Threshold: bi-directional flows

Figure 6.19 displays the number of bi-directional flows. The figure illustrates that the daily pattern of the number of flows outweighs the variations caused by the attack periods. Nonetheless, there is a slight increase observed during the ping flooding attack. It is worth noting that the pattern observed in Figure 6.19, which indicates a decrease in the afternoon, may not be as pronounced when considering the overall weekly or daily pattern, as depicted in Figure 5.11.

In the study by Zhou et al. [21], the researchers ranked the number of bi-directional flows as the 20th feature based on its detection performance. The outputs appear to align with this ranking, supporting the paper's accuracy.

6.3.8 Entropy and entropy rate: packet size

The entropy and entropy rate of packet sizes were among the few features that could be extracted from both the NetFlow data set and the router interface telemetry data set, allowing for a comparison. In theory, these two sets should exhibit similar patterns. However, they appear quite different perhaps due to the asymmetry in the interfaces used for capturing traffic in the two data sets. Figure 6.20 illustrates the

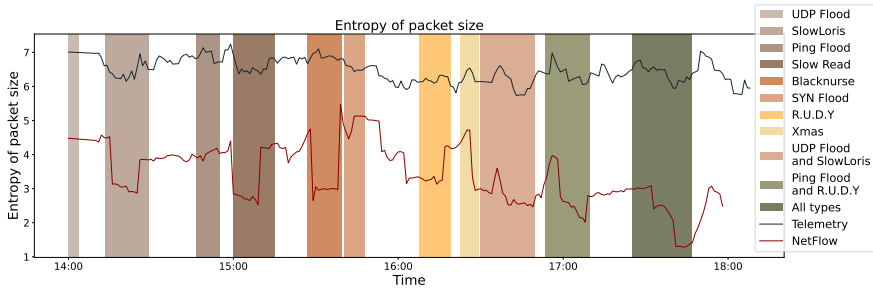


Figure 6.20: The entropy of packet sizes for attack generation round 3 on AR1

entropy of the packet size distribution during the third round of attack generation. The entropy is calculated using Equation (2.8), where $P(X)$ represents the probability distribution of packet sizes as defined in Equation (5.7), and α is set to 10. The entropy in Figure 6.20 is computed over a 10-minute window with a frequency of one minute.

Figure 6.20 shows that the entropy values of the two data sets do not have the same baseline. This discrepancy can be attributed to both the asymmetry in the interfaces used for data collection and the fact that the router interface telemetry data set does not have direct access to the precise packet size values. Packet sizes are derived from the bytes and packets per second measurements in the telemetry data set. In contrast, the NetFlow data set provides more accurate packet sizes by directly including the bytes and packets per flow. However, there are minor fluctuations in the entropy of the telemetry data set that align with corresponding changes in the NetFlow data set. Notably, during the SlowLoris attack and the Slow Read attack, a significant decrease can be observed in entropy in the NetFlow data set, accompanied by a slight decrease in the telemetry data set. Additionally, both data sets show an increase in entropy after the conclusion of the final attack.

The entropy rate of the packet size distribution was computed using Equation (5.8), where the entropy is defined as described earlier. Figure 6.21 illustrates the disparity in entropy rates between the two data sets. In the paper [21], the authors suggest that the number of different packet sizes increases during a DDoS attack to evade detection. They calculate the entropy rate of the packet sizes to observe a decreasing trend throughout all types of attacks. However, the ground truth generation did not specifically employ attacks with random packet sizes, although the combination attacks could be the closest example of random sizes.

From Figure 6.21, one can observe that the entropy rate of the NetFlow data

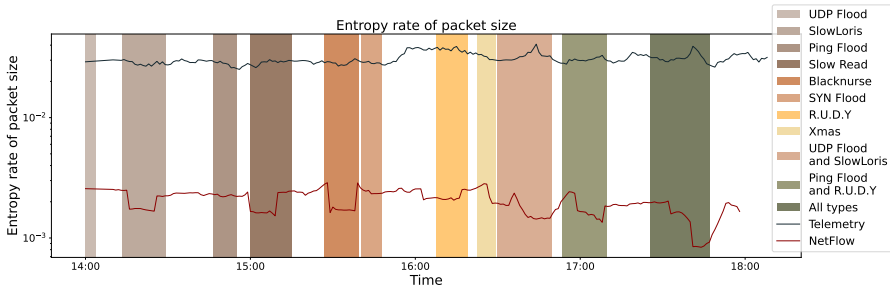


Figure 6.21: The entropy rate of packet sizes for attack generation round 3 on AR1

follows a similar fluctuation pattern to that of the entropy since there were no launched attacks with random packet sizes. However, the fluctuations in the entropy rate are dampened compared to the entropy values because the entropy is divided by the number of different packet sizes. On the other hand, the telemetry data does not consistently exhibit the same fluctuation pattern. For instance, during the combination of the UDP flood and SlowLoris attack, the entropy in Figure 6.20 decreases towards the end while the entropy rate increases. According to Equation (5.8), this indicates a significant decrease in the number of different packet sizes. During a flooding attack, a decrease in different packet sizes is expected because the flood of packets typically has a uniform size. The reason why this is not reflected as prominently in the NetFlow data might be because it utilizes more accurate packet size information, making the entropy more sensitive to changes. When considering the average packet size in the telemetry data, the changes are never as substantial as in the NetFlow data. Consequently, a decrease in different packet sizes leads to an increase in entropy rate because the entropy does not increase proportionally.

Overall, the differences in the entropy rate patterns between the two data sets might be attributed to the varying levels of precision in packet size measurements and the impact it has on entropy calculations.

6.3.9 Entropy and entropy rate: destination IP addresses

The entropy of destination IP addresses can provide insights into changes in the probability distribution of addresses within a specific time window. To calculate the entropy, the generalized entropy equation (Equation (2.8)) is employed, where $P(X)$ represents the probability distribution of destination IP addresses (as defined in Equation (5.9)), and α is set to 10. Figure 6.22 displays the variations in entropy of destination IP addresses over time, using a sliding window size of 10 minutes that is shifted every minute. By utilizing the generalized entropy with an α value

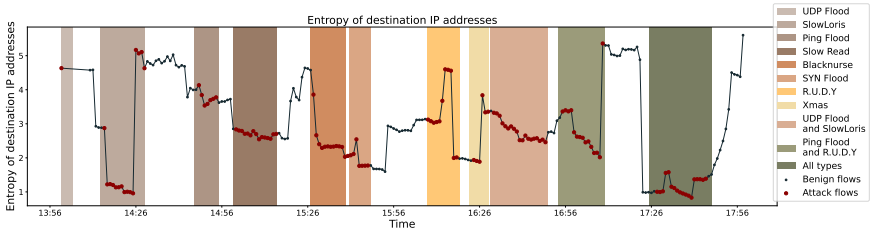


Figure 6.22: The entropy of destination IP addresses on AR1

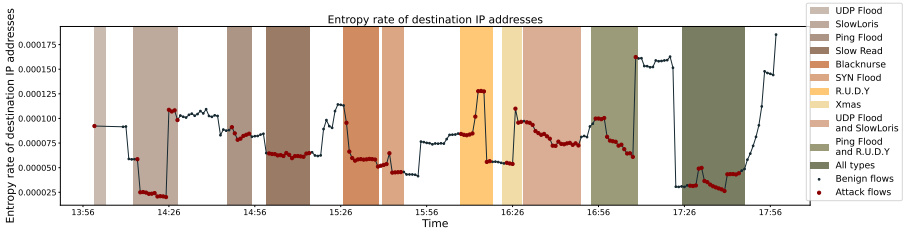


Figure 6.23: The entropy rate of destination IP addresses on AR1

of 10, even small changes in entropy should become noticeable. Figure 6.22 shows considerable fluctuations in entropy. While it is somewhat challenging to determine if the entropy is affected by the generated attacks, there seems to be a significant decrease during the Blacknurse attack. Additionally, the entropy shows a notable increase immediately after the combination of a ping flood and a R.U.D.Y attack. Higher entropy implies more uncertainty in the destination IP addresses, so an increase in entropy after an attack is reasonable since the victim's address is no longer overrepresented in the probability distribution.

Equation (5.10) was utilized to calculate the entropy rate. An example of the entropy rate of destination IP addresses can be observed in Figure 6.23. The plot follows a similar pattern to the entropy plot but on a significantly smaller scale. Since the entropy rate is calculated by dividing the entropy by the number of different destination IP addresses within the time window, it is logical for them to have the same pattern. During the attacks, the attack packets count towards one destination IP address, which adds only one additional number to the divisor in the entropy rate equation. This leads to the form being the same as the entropy.

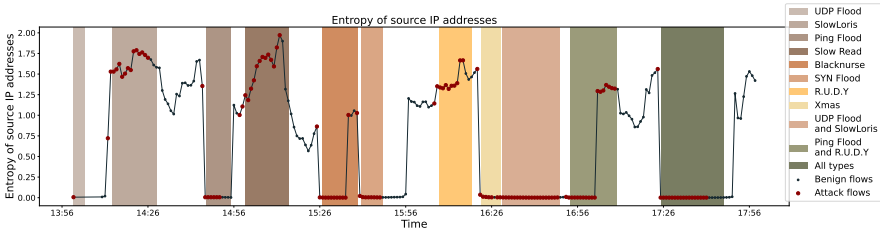


Figure 6.24: The entropy of source IP addresses on AR4

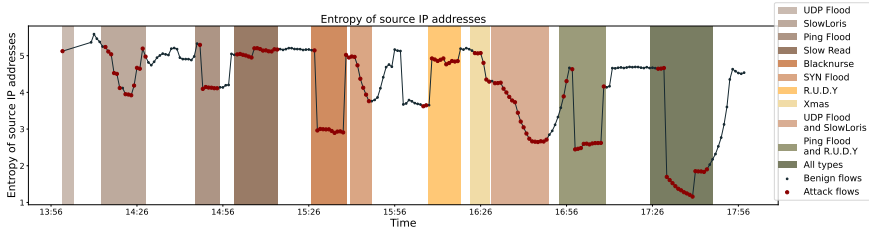


Figure 6.25: The entropy of source IP addresses AR2

6.3.10 Entropy and entropy rate: source IP addresses

The calculation of entropy and entropy rate for source IP addresses follows the same methodology as that for destination IP addresses. Analyzing the changes in the entropy of source IP addresses over time can provide insights into the distribution of hosts generating traffic. In routers with a limited number of monitored interfaces, the entropy will naturally be lower than in routers with more monitored interfaces, as the traffic may originate from a smaller pool of hosts.

Figure 6.24 illustrates the entropy of source IP addresses over time in a router that collects NetFlow data from only a few interfaces. In this case, the entropy never exceeds the value two due to the limited source address pool resulting from monitoring a small number of interfaces. Contrarily, in the router AR2 connected to an IXP, depicted in Figure 6.25, the entropy typically falls between four and six, with rare instances of dropping towards two. When the entropy does decrease to two, it could be considered an anomaly.

The researchers in [21] proposed utilizing entropy rate to detect LRDDoS attacks. They argue that during flooding DDoS attacks, the entropy of source IP addresses increases as the number of source IP addresses in the distribution also increases. However, there was a limited number of attackers in the generated attacks, resulting

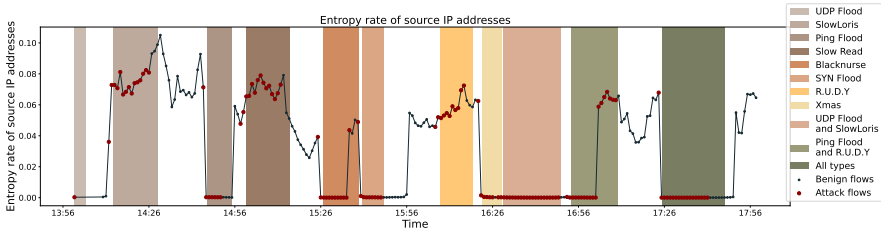


Figure 6.26: The entropy rate of source IP addresses on AR4

in the same number of source IP addresses during all attacks. Therefore, the number of source IP addresses does not increase significantly across different attacks.

Figure 6.26 displays the entropy rate of the same router as shown in Figure 6.24. The plots exhibit similar patterns, except after the SlowLoris attack, where the entropy rate increases while the entropy decreases. As mentioned in Section 6.3.8, this could be attributed to a decrease in the number of different source IP addresses after the attack. It is logical to observe a decrease in different source IP addresses as the distribution no longer includes the attacker addresses. However, it indicates that the entropy did not decrease proportionately with the number of different addresses. This behavior aligns with the nature of the SlowLoris attack, which is a LRDDoS attack and may not generate a significant number of packets to significantly increase the attacker IP addresses proportion in the distribution.

6.3.11 Entropy and entropy rate: bi-directional flows

Entropy and entropy rate for bi-directional flows was calculated using the generalized entropy equation, similar to the calculations for other features. Figure 6.27 shows the entropy of bi-directional flows. From the figure, one can observe that the entropy of bi-directional flows fluctuates throughout the day. However, the most significant changes in entropy are not caused by the generated attacks. This observation aligns with the behavior of the entropy of destination IP addresses. The researchers in [21] ranked this feature as the 21st out of 27 features, indicating its limited usefulness. Figure 6.27 seems to lead to a similar conclusion, as the entropy even increases at the beginning of the SYN flood attack. This increase implies that the distribution of bi-directional flows becomes more random due to the attack. This development can be attributed to the introduction of five new bi-directional flows into the distribution, which increases the value of n in the entropy formula. This increase was why [21] proposed using the entropy rate of bi-directional flows, which they ranked as the second-best feature. The output of the calculations on the entropy rate can be seen in Figure 6.28, which was found using Equation (5.12).

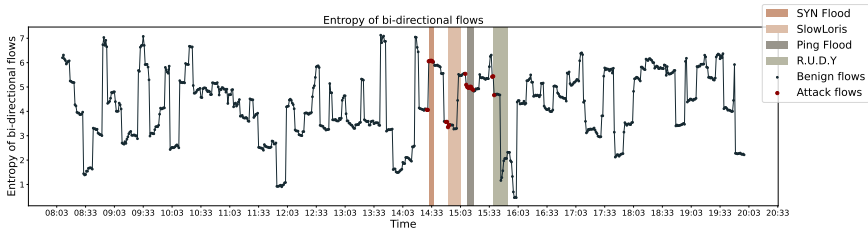


Figure 6.27: The entropy of bi-directional flows on AR1 during attack generation round 1

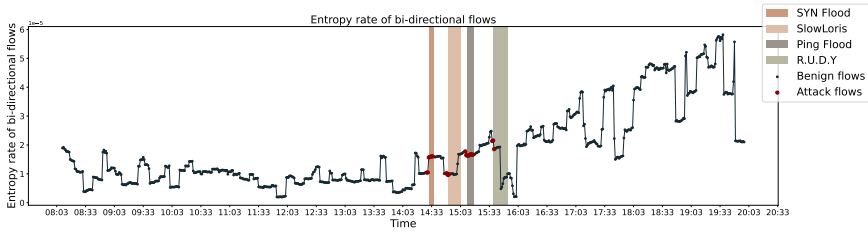


Figure 6.28: The entropy rate of bi-directional flows on a AR1 during attack generation round 1

From Figure 6.28, it appears that the entropy rate of bi-directional flows is not significantly affected by the generated attacks. The fluctuations are dampened, but the overall pattern of the plots remains the same. The entropy rate still increases during the SYN flood attack, indicating that the entropy increases more than the number of bi-directional flows. This is likely because the attacks do not introduce many additional bi-directional flows, but they account for a prominent proportion of the packets regardless.

During periods of decreased entropy, such as during the R.U.D.Y attack and ping flood, the decrease in entropy rate is less prominent, making it challenging to utilize for detection purposes. If the attacks involved more attackers, it might have yielded different results.

6.3.12 Entropy: SYN flows

The analysis was extended to calculate the entropy of different features specifically for SYN flows, aiming to gather more information about potential SYN flood attacks. The focus is on three features of the SYN flows: the entropy of destination IP addresses, source IP addresses, and bi-directional flows.

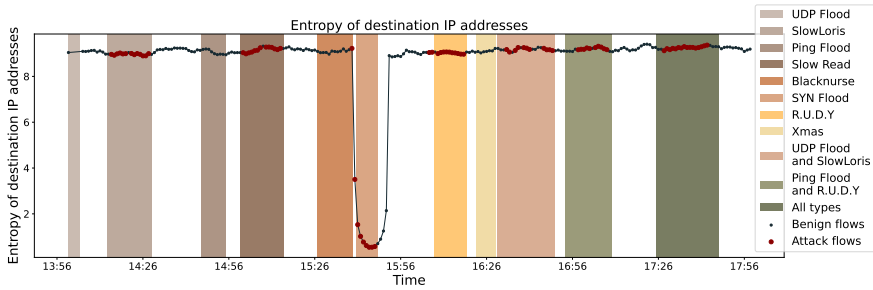


Figure 6.29: The entropy of destination IP address flows with the SYN flag set

Destination IP addresses In Figure 6.29, the entropy of destination IP addresses in SYN flows during the third round attack generation is depicted. The entropy is calculated using a sliding window technique with a window size of five minutes. The figure shows a noticeable decrease in entropy during the SYN flood attack, while the entropy remains relatively stable at a high level during other periods. This decrease in entropy aligns with what is expected, indicating that the destination IP addresses in SYN flows become less random during the SYN flood attack.

However, when all attack types are combined, one could not observe the same decrease in entropy as expected. There could be several reasons for this. One possibility is that due to the simultaneous execution of all attacks, the attack machines might have faced limitations in generating all the attack packets. Consequently, some attack packets might not have been produced, leading to an incomplete representation of the attack flows in the collected NetFlow records.

Another factor that could contribute to the observed discrepancy is the sampling technique used by the router to collect NetFlow records. Since each router only samples one out of every 100 packets, the SYN flood flows could be among the other 99 packets. Therefore, the collected NetFlow records might not fully capture the SYN flood flows, which could explain the limited decrease in entropy.

The figure shows that the router collected some attack flows during the last attack period, but they could belong to the LRDDoS attacks and would not render a large decrease in entropy. Furthermore, the specific router shown inFigure 6.29, AR2, is connected to an IXP. As a result, the stability of the entropy and its lack of significant impact from the attack flows during the last attack period could be attributed to the nature of the traffic entering this router. Traffic coming into the router from another network destined for customers within Sikt's network will likely have a more consistent set of destination IP addresses than packets destined to hosts

outside Sikt's network. This stability in the destination IP address distribution leads to a more stable entropy, making it less susceptible to fluctuations caused by the attack flows.

In contrast, other routers with a lower base-rate entropy exhibited a decrease in entropy during the last attack and the LRDDoS attacks. The decrease in entropy was also most pronounced during the SYN flood attack in these routers.

Source IP addresses The pattern observed for the entropy of source IP addresses exhibited some differences. Figure 6.30 showcases that the entropy tends to remain consistently low, except during the attack periods. This could be attributed to the fact that the traffic originates from customers connected to this particular router, resulting in a limited address pool with a relatively stable distribution.

Interestingly, during the Blacknurse attack, one can observe an increase in entropy that does not appear to be caused by the attack flows in the ground truth generation. This could be another anomaly as the overall pattern seems to remain stable. However, when calculating the entropy with a larger window size, the spike diminishes to a value of 0.25. In comparison, the spike during the SYN flood attack becomes more prominent. This suggests that the increase in entropy during the Blacknurse attack was due to a sudden influx of packets from a specific source IP address, which quickly subsided.

A similar pattern emerges for the increase in entropy resulting from the UDP and SlowLoris attacks. With the larger window size, the increase only had a value of 0.45. This could be attributed to when all the connections of the SlowLoris attacks are established, which explains why the entropy quickly decreases when the connections are established.

Overall, the entropy of source IP addresses remains relatively low and stable, except during some attack periods, where specific events lead to temporary increases.

Bi-directional flows The last calculation done on the SYN flows was to calculate the entropy of bi-directional flows. The pattern of this metric is sometimes almost identical to the entropy of the destination IP addresses. In the paper [21], the researchers excluded the entropy of destination IP addresses as a metric because they claimed that the feature was redundant with the entropy of bi-directional flows. This is not always the case for all of the routers. From the analysis done in this thesis, it seems that if the router has a stable high entropy of destination IP addresses, and a stable low entropy of source IP addresses, the entropy of bi-directional flows will generally be high. Yet, the effect of the generated attacks will be more prominent. An

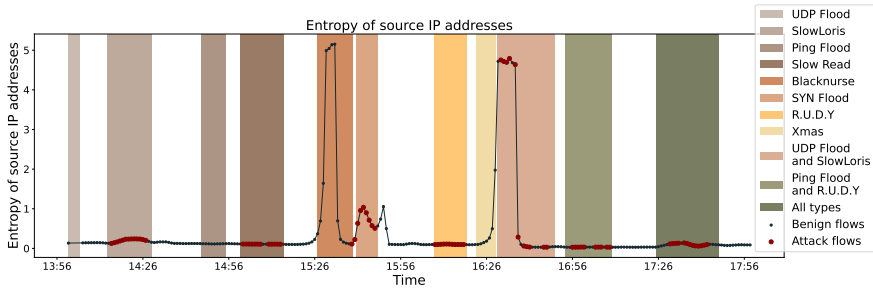


Figure 6.30: The entropy of source IP address flows with the SYN flag set

example of this can be seen in Figure 6.31, which is the same router from Figure 6.29. For this router, there did not seem to be a clear impact on the entropy of destination IP addresses resulting from attacks other than the SYN flood attack. Yet, if the entropy of bi-directional flows was used, the impact of all the attacks involving SYN packets was visible in the figure. Even though the impact of the last attack is more pronounced in the entropy of bi-directional flows, it is still not as apparent as the SYN flood attack. This could lead to the belief that the SYN flood packets were either not generated or sampled away.

The last calculation done on the SYN flows was to calculate the entropy of bi-directional flows. In some cases, the pattern of this metric closely resembled that of the entropy of destination IP addresses. However, in the paper by Zhou et al. [21], the researchers excluded the entropy of destination IP addresses, considering it redundant with the entropy of bi-directional flows. Nevertheless, this is not always the case for all routers.

From the observations made in this analysis, it seems that when a router exhibits a stable and high entropy of destination IP addresses, along with a stable and low entropy of source IP addresses, the entropy of bi-directional flows generally tends to be high. However, the impact of the generated attacks becomes more pronounced. An example of this behavior can be observed in Figure 6.31, which depicts the same router as Figure 6.29. In this case, the impact on the entropy of destination IP addresses resulting from attacks other than the SYN flood attack was not noticeable. However, considering the entropy of bi-directional flows, the impact of all attacks involving SYN packets becomes visible. Although the impact of the last attack is more evident in the entropy of bi-directional flows, it is still not as prominent as the SYN flood attack. This leads to the speculation that the SYN flood packets were not generated or sampled away. For other routers that did not have such a stable source and destination IP entropy, the entropy of bi-directional flows was identical to

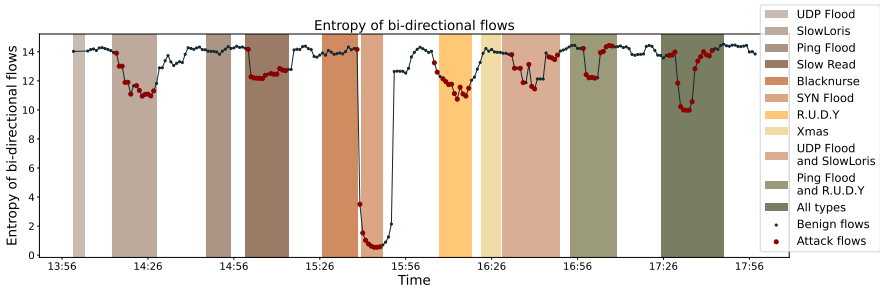


Figure 6.31: The entropy of bi-directional flows with the SYN flag set

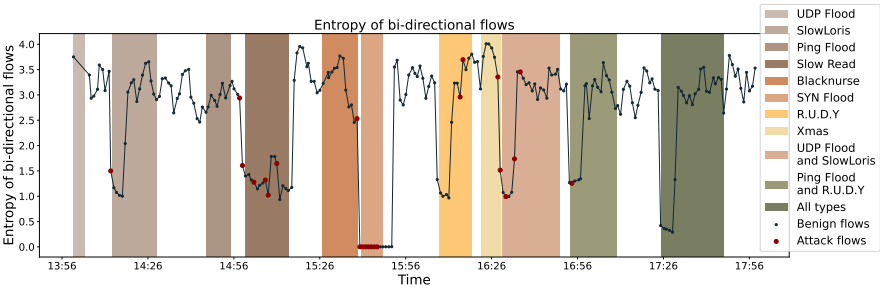


Figure 6.32: The entropy of bi-directional flows with the SYN flag set

that of the destination IP entropy. If both entropy measures varied in their behavior, it could be seen that the entropy of bi-directional flows was easier influenced by both in different directions.

If the source and destination IP entropy had opposite behaviors, like one being stable and the other varying a lot, the entropy of bi-directional flows was very different from the entropy of destination IP addresses. This makes sense as the bi-directional flows are source and destination IP address pairs, so if, e.g., the source IP address distribution is stable while the destination IP address distribution varies a lot, the distribution of bi-directional flows will vary even more because as the same customer sources use different destination IP addresses a new bi-directional flow is made. This variation might make distinguishing between false and true positives hard.

6.3.13 Top 20 destination IP address flows

A different metric was calculated to assess the change in the top 20 flows based on packets for each destination IP address every minute. The comparison was

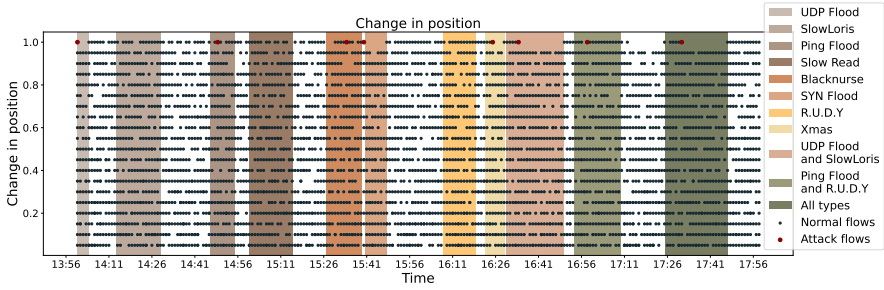


Figure 6.33: The change in the top 20 flows on AR3 for attack generation round 3

made between the 20 largest flows in the current minute and the previous minute, determining if there were any changes in the top positions. An alert was generated when a flow appeared in the current top flows but was absent in the previous minute's top flows or when a flow moved up in position. The change in position was measured as a percentage using Equation (5.13), with a value of 1.0 or 100% indicating that the flow changed into the top position in the current minute from being absent in the previous minute. Analyzing Figure 6.33, all the changes in the top 20 flows based on destination IP addresses during the attack period can be observed. Changes in the top 20 positions occurred frequently, and the attacks did not appear to influence the frequency. However, in some attack periods involving attack flows, there were fewer instances of 100% changes in position after the attack flow arrived. This was particularly noticeable in certain high-volume attacks, such as the UDP floods, SYN flood, and the combination of all attacks. One possible explanation is that the victim IP address remained in the top position for an extended duration, preventing other flows from achieving a 100% change in position. The router in Figure 6.33 only recorded the attack flows when they experienced a 100% change, likely because smaller attacks, such as the LRDDoS attacks, could not compete with the overall traffic passing through the router in terms of size. Other routers with lower traffic volumes consistently displayed the attack flows in the top 20 positions for all types of attacks. An example of such a router, AR4, can be observed in Figure 6.34. In the case of LRDDoS attacks, some instances of 100% changes were observed. As the attack progressed, lower percentage changes were noted, indicating that flows entered the top 20 at lower positions or climbed up to higher positions. Logically, during the initial phase of LRDDoS attacks, the attack flows experienced a 100% change as numerous HTTP connections were established simultaneously to exhaust the server's resources by keeping these connections open. Once the connections were established, fewer packets were required to maintain them open, resulting in minor changes in position. The lower traffic load on the router allowed for the detection of LRDDoS attacks.

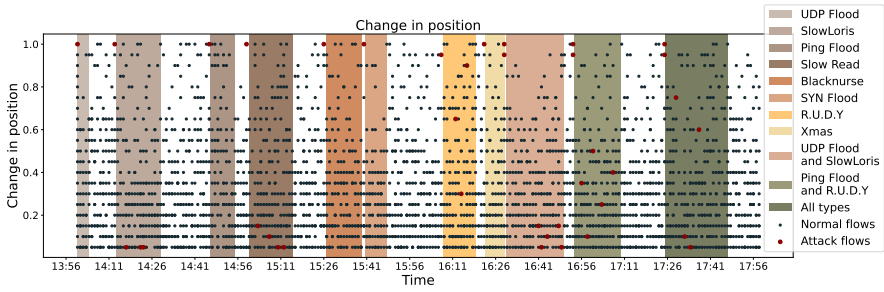


Figure 6.34: The change in the top 20 flows on AR4 for attack generation round 3

LRDDoS attacks were also evident in the top 20 flows of the router closest to the victim, VR. The attack flows in this router belong to the attacker's IP addresses. Why the attacks could be seen in the router could be attributed to its lower incoming traffic load. However, it may also be due to the victim producing packets during the LRDDoS attack in an attempt to respond to the attackers' requests. Ideally, no packets should be received from the victim, as it should be unavailable if the attacks have achieved their intended objective.

6.3.14 Random Forest

A RF classifier was trained on eight different feature sets, which are explained in Section 5.3.1.

Fields One of the feature sets consisted of a selection of NetFlow header fields. The RF classifier was trained using this feature set with and without IP addresses to investigate whether the classifier would solely detect attack flows due to the presence of the same IP addresses in the training and testing sets. Comparing Figure 6.35 and Figure 6.36 illustrates the difference between training the classifier with and without IP addresses. The classifier trained without IP addresses appeared to have more false positives. This result aligns with the expectation since the same IP addresses were used for the attackers in both the training and testing data.

Additionally, the classifier missed some alerts in Figure 6.36 compared to Figure 6.35. These missed alerts correspond to flows belonging to attacks that the classifier was not trained on. It is expected that supervised learning algorithms have difficulty detecting unknown attacks. The only exception was during the Slow Read attack, where the classifier without IP addresses detected some attack flows. This could be a coincidence, as it also produced many false positives in general, but it prevented the attack from going undetected.

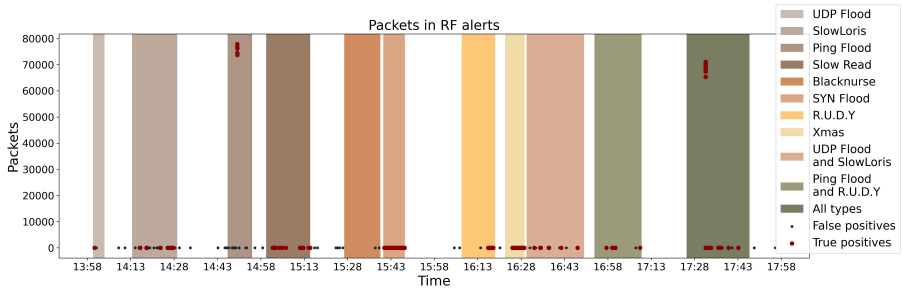


Figure 6.35: The number of packets in each alert from a Random Forest classifier run on attack generation round 3 using the header fields as the feature set

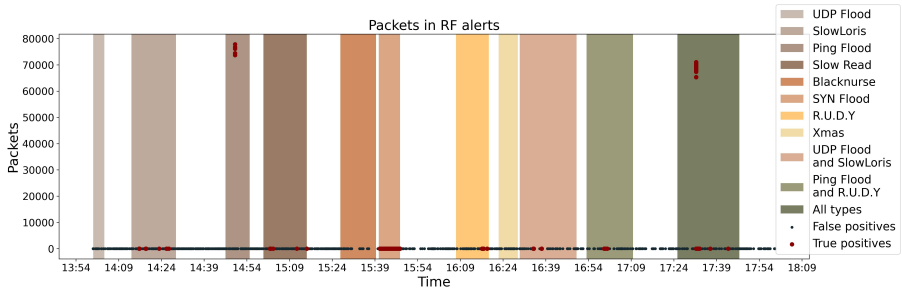


Figure 6.36: The number of packets in each alert from a Random Forest classifier run on attack generation round 3 using the header fields as the feature set without IP addresses

The RF classifier was also trained on router interface telemetry fields: queue size, ingress and egress packets, and bytes per second. Labeling the attack data for the router interface telemetry sets was less precise than with the NetFlow data since IP addresses were not available. Instead, all interface telemetry measurements during attack periods were labeled as malicious. However, mislabeling occurred, particularly during periods of inactivity, leading the classifier to label them as malicious due to being within the attack period. This mislabeling likely affected the classifier's performance in accurately labeling potential attacks. Despite this, training the RF classifier on router interface telemetry data had the advantage of providing data from all interfaces, not just customer interfaces. Having data from all interfaces is advantageous for detecting and identifying attacks more comprehensively and at an earlier stage.

Additionally, collecting telemetry data every two seconds made it less resource-

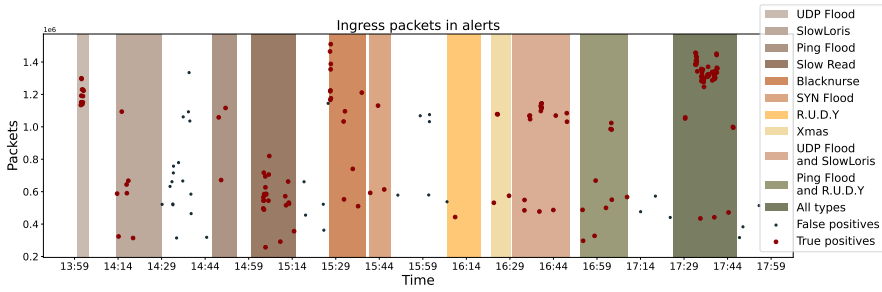


Figure 6.37: The number of packets in each alert from a Random Forest classifier run on attack generation round 3 using the router interface telemetry fields as the feature set

intensive to calculate various metrics than NetFlow data, which could involve thousands of flows per second. Furthermore, the telemetry data takes less space, simplifying storage compared to NetFlow records. One possible solution could be to sample the flows less often, but this would also make the data less fine-grained. One possible solution to mitigate the decrease in granularity caused by sampling flows less frequently could be for the ISP to collect NetFlow records on more interfaces, compensating for the information loss. However, this is a trade-off between resources and detection accuracy. The network managers believe that the NetFlow records collected from all incoming customer interfaces and IXP connections contain all the necessary information. Nonetheless, some attacks, such as LRDDoS attacks, were not visible in more trafficked routers but might have been in some core routers.

In Figure 6.37, alerts from the RF classifier in the core router CR4 are displayed. Despite being positioned in the middle of the attack route, most alerts fell within the attack periods. The classifier produced alerts for all types of attacks without excessive alerts outside the attack periods. However, the drawback of having a low number of false positives was a significant number of false negatives. Since the measurements were collected every two seconds, there should ideally have been more alerts during the attack periods. However, when performing alert fusion, any contribution to the holistic view can be helpful, reducing the significance of missing some alerts. Even though it might not detect all the malicious traffic, it still manages to alert at least once about every attack. This could be useful because the network managers are not overwhelmed with alerts they must process. The trade-off between false positives and false negatives always exists, each with advantages and disadvantages.

Entropy Another RF classifier was trained on entropy and other features extracted from the header fields of NetFlow records. A paper [21] inspired this feature set,

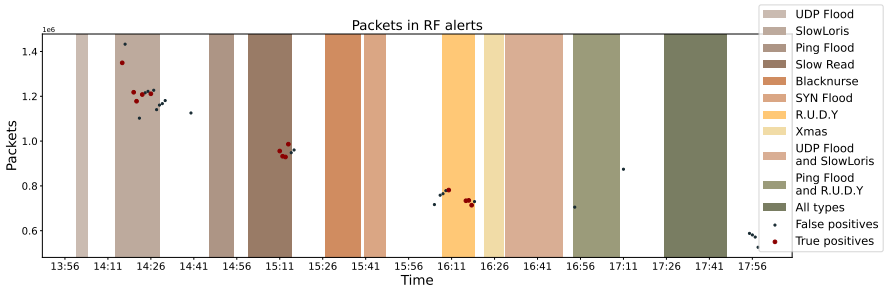


Figure 6.38: The number of packets in each alert from a Random Forest classifier run on attack generation round 3 using entropy and other metrics as the feature set

which achieved promising results using some of these features to train a RF classifier. However, the results in this thesis did not demonstrate the same level of promise. Figure 6.38 illustrates the alerts generated by the classifier in router AR3, which is closest to an attacker. Although the classifier did not produce many alerts overall, the ones it did produce were not always accurate. It is important to note that the labeling of traffic in the testing dataset was not 100% accurate due to the nature of the metrics used. For example, the entropy of destination IP addresses was calculated using a sliding window technique, considering multiple flows for each measurement. Therefore, a measurement was labeled as malicious if there was at least one attack flow between the measurement and the previous measurement. As the sliding window moved forward every minute, any minute containing an attack flow was labeled as malicious. This allowed some margin for error in the classifier's predictions.

The alerts primarily originated from LRDDoS attacks, which was one of the objectives of [21] to detect. The classifier also alerted on the Slow Read attack, an unknown attack to the classifier. Varying the sliding window size was explored, and increasing the window size led to alerts disappearing during the SlowLoris attack period. During the 10 and 15-minute windows, all alerts on the Slow Read attack disappeared. However, more alerts were observed during the R.U.D.Y attack, and even a few at the end of the ping flood and combined R.U.D.Y attack. Since these attacks were present in the training set, it may explain why they continued to trigger alerts when others did not.

For router interface telemetry data, a RF classifier was also trained on entropy metrics. Figure 6.39 displays the same router trained on the entropy metrics of the telemetry data. This classifier generated far more alerts, covering all types of attacks, including unknown ones. Some false positives were observed between attack periods, but the true positives seem to outweigh them.

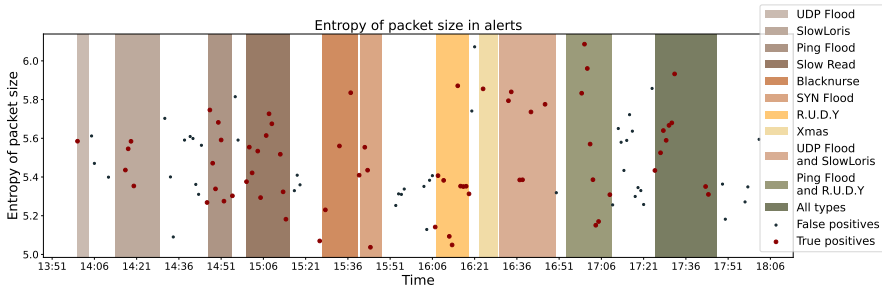


Figure 6.39: The number of packets in each alert from a Random Forest classifier run on attack generation round 3 using entropy metrics on router interface telemetry data as the feature set

Altering the window size for this feature set resulted in a decrease in the number of alerts for the 10-minute window, followed by a more significant increase in the 15-minute window. This increase was particularly evident during combined attack periods. It could be explained by the increased randomness of packet sizes in combined attacks, where multiple attack packet types are sent. This increase in the entropy of packet sizes might have been used as a triggering factor by the classifier. Although the classifier was not trained on any combined attacks, its ability to detect them was impressive. The closest behavior observed in the training set was during the R.U.D.Y attack. The larger window size might provide a more stable entropy value, and the classifier might consider a sudden decrease in entropy rate anomalous.

Combined The last category of feature sets involved combining entropy metrics and fields to gain additional insights into the attack situation. Figure 6.40 presents the alerts generated by the classifier trained on the combined feature set. Although the number of alerts was low, they were all accurate. The classifier was also trained on the same feature set without IP addresses, resulting in more false positives and the loss of true positives, except during the SYN flood attack. However, there was a new true positive during the SlowLoris attack. Most false positives occurred during the periods of LRDDoS attacks. This behavior resembled that of the classifier trained on the entropy data set.

The classifier depicted in Figure 6.40 successfully identified the Blacknurse attack, which was unknown to it. However, other unknown attacks went undetected. The classifier without IP addresses lacked this ability, suggesting the attack was detected because it used the same attacker IP as in the training set. Increasing the window size led to the classifier without IP addresses producing no accurate alerts.

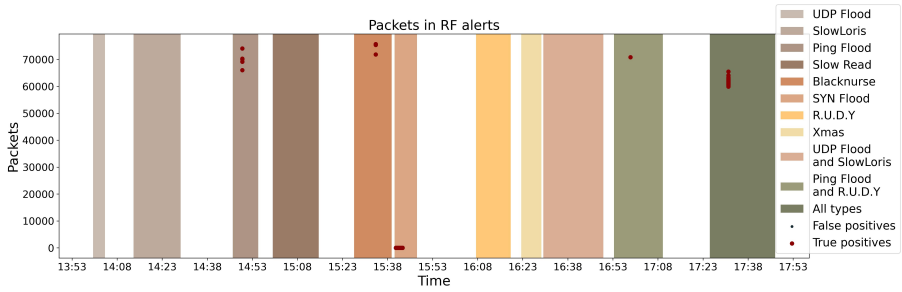


Figure 6.40: The number of packets in each alert from a Random Forest classifier run on attack generation round 3 using entropy metrics and header fields as the feature set

On the other hand, the classifier with IP addresses exhibited increased alerts during the ping flood and the final attack but lost the true positives during the SYN flood attack. The instability in the number of alerts and the overall lack of alerts on attacks may indicate poor feature selection for this particular feature set. Supervised algorithms for feature selection could have helped mitigate this issue, but as access to ground truth data was obtained after developing the detection methods, it was not employed. Future work on the topic could explore this aspect further.

Numerous alerts were observed regarding the combination of router interface telemetry features, as depicted in Figure 6.41. Although there were several false alerts, all attacks were alerted on. Increasing the window size resulted in changes in the number of alerts. With a 10-minute window, alerts during the SlowLoris attack disappeared, most alerts during the SYN flood attack vanished, and alerts at the beginning of the ping flood attack were no longer generated. This behavior is somewhat unexpected, considering these three attacks were part of the classifier's training set. The same behavior was observed in the feature set with only entropy, suggesting that it might be attributed to this component of the feature set. When a 15-minute window is used, the alerts reappear, except for those during the end of the SlowLoris attack. As the SlowLoris attack lasts for approximately 15 minutes, at the end of the attack, the entropy calculation would predominantly involve traffic from the attack period. This might yield a more stable entropy value, which the classifier might not perceive as anomalous.

6.3.15 K-means

The unsupervised learning algorithm K-means was implemented as the final detection method in the study. Three feature sets were considered for each data set, resulting in six combinations. The clustering process was performed every 15 minutes for

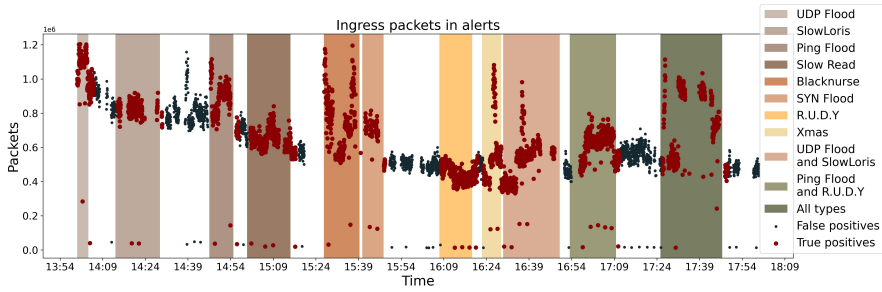


Figure 6.41: The number of packets in each alert from a Random Forest classifier run on attack generation round 3 using entropy metrics and router interface telemetry fields as the feature set

data sets using window sizes of 5 and 10 minutes and every 30 minutes for data sets using a 15-minute window size. The decision to cluster the traffic at regular intervals was motivated by the large number of NetFlow records, which exceeded the processing capacity of Python for simultaneous analysis. Although alternative Python modules could have been explored to address this limitation, such investigation was not prioritized. The choice of a 15-minute clustering interval was based on the observation that most DDoS attacks conclude within this timeframe [1].

Fields The K-means clustering algorithm was applied to the same header fields used in the RF classifier for analyzing the NetFlow data. The clustering results are presented in Figure 6.42. It should be noted that the router depicted in the figure exhibits relatively low traffic volume, leading to a lower number of flows compared to other routers. The clustering algorithm appears to successfully group some attack flows into the appropriate cluster, although many attack flows are also present in the other cluster. Notably, during UDP and SYN flood attacks, attack flows are consistently grouped correctly throughout the entire attack duration. However, there are instances where attack periods appear to have been assigned to the cluster labeled as “normal”.

The two clusters were labeled using the algorithm proposed in [61]. The study establishes thresholds to differentiate between periods with massive attacks and without and determine how big the difference in cluster diameter should be. In this thesis, a threshold of 0.5 was adopted for the Davies-Bouldin score, indicating that scores below this threshold were associated with massive attacks. Regarding the cluster diameter thresholds, any observed differences were deemed relevant.

Evaluating the Davies-Bouldin scores for each clustering process may provide valuable insights into the observed patterns in Figure 6.42. The highest Davies-

Bouldin score was recorded during the ping flood attack, reaching a value of 0.02. According to the threshold, this would suggest the presence of a massive attack, aligning with the nature of a ping flood. However, higher Davies-Bouldin scores generally indicate suboptimal clustering, implying challenges in effectively separating the traffic into two distinct clusters. An intriguing observation is found during the SlowLoris attack, which exhibited the lowest Davies-Bouldin score of 0.0085. This outcome is somewhat unexpected since clustering the flooding attack, characterized by identical packet sizes and protocols, would seemingly be easier compared to LRDDoS attacks that involve different sizes of HTTP requests. The low score of 0.0085 indicates the presence of a massive attack, prompting the cluster labeling algorithm to assess the cluster diameters. In this scenario, the attack cluster should exhibit the smallest diameter due to its homogeneous traffic. However, the algorithm fails to label the cluster during the SlowLoris attack correctly. Although the labeled attack cluster's diameter is 2.5 times more compact than the other cluster, the SlowLoris attack, which is not a massive attack, has a higher diameter, resulting in the cluster containing most attack flows being labeled as "normal".

Surprisingly, none of the Davies-Bouldin scores exceed the threshold for this router, which contradicts expectations, especially during attack-free periods. It is possible that the cluster labeling algorithm is not well-suited for this particular router due to its specific traffic patterns. With only a few customer interfaces, the traffic may exhibit significant similarity, which may explain the consistently low Davies-Bouldin scores. Most flows are clustered into the "attack" cluster regardless of the time. The only exception occurs during the Slow Read attack, where the Davies-Bouldin score still suggests a massive attack. However, the larger cluster is 1.5 times less compact than the cluster containing Slow Read flows, thereby selecting it as the attack cluster.

The most accurate cluster labeling appears to be achieved during the UDP flood and SlowLoris attacks, where false positive flows are minimal, and a substantial number of true positive flows are detected. In this case, the Davies-Bouldin score reached 0.01, with the attack cluster being 11 128 times more compact than the "normal" cluster. This finding appears somewhat peculiar regarding a combination attack, which should not generate identical flows and consequently exhibit reduced compactness. However, during the Xmas attack, the attack cluster demonstrates even higher compactness than the "normal" cluster, suggesting that the SlowLoris flows might be clustered into the "normal" cluster during this combination attack.

Regarding the telemetry data set, the behavior was somewhat different. Figure 6.43 presents the results of the K-means clustering performed on the same router as depicted in Figure 6.42. The clusters are visualized using red and black dots. Due to the absence of IP addresses for accurate labeling, measurements during attack

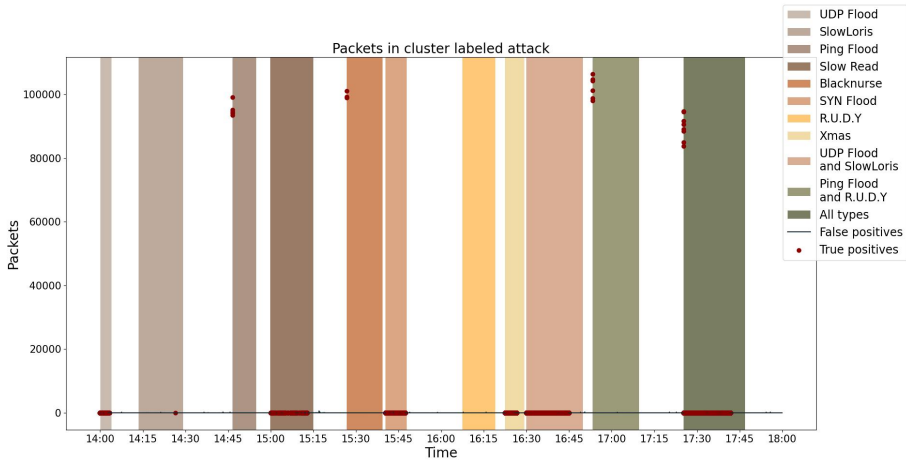


Figure 6.42: The number of packets in the attack cluster on attack generation round 3 using NetFlow header fields as the feature set

periods were assumed to belong to the generated ground truth data.

Interestingly, between the Slowloris attack and ping flood, a significant portion of the traffic is erroneously labeled as attack traffic. However, the Davies-Bouldin score for this clustering process is 0.36, indicating a massive attack. Analyzing the cluster diameter measurements reveals that the "attack" cluster is nearly twice as compact as the "normal" cluster. This observation could be attributed to the nature of router interface telemetry measurements, which only reflect the traffic load during specific time intervals. If the traffic load remains stable, the clustered measurements will likely exhibit similarities, leading to a reduced cluster diameter.

In contrast to the NetFlow clusters, Davies-Bouldin scores exceeding the threshold of 0.5 are encountered in the telemetry data set. Notably, the ping flood and R.U.D.Y attack exhibit the highest score of 0.75, with the measurements evenly split between the clusters. This outcome may be attributed to an attack comprising two distinct attack vectors, which complicates the clustering process by simultaneously clustering them together and differentiating them from benign traffic.

The lowest Davies-Bouldin score of 0.23 is recorded during the Xmas attack and some of the R.U.D.Y attack. Here, the attack cluster is only 1.5 times as compact as the normal cluster; nevertheless, the clustering appears satisfactory based on the Davies-Bouldin score. Furthermore, it is worth noting that some measurements during the intervals between these two attacks are also mislabeled as attacks, which

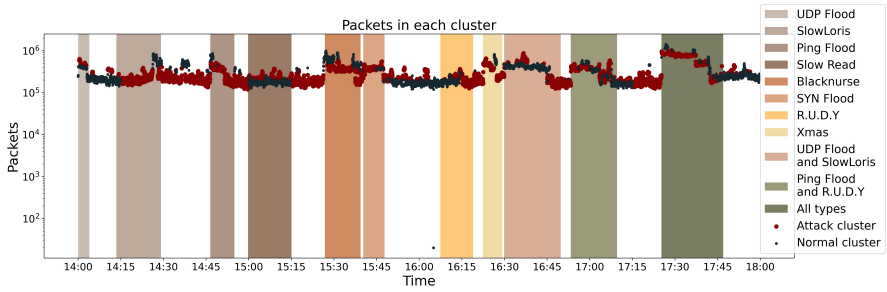


Figure 6.43: The number of packets in the attack cluster on attack generation round 3 using router interface telemetry fields as the feature set

is an incorrect classification. Unfortunately, similar misclassifications can be observed in almost all the breaks between attacks.

Entropy The second feature set subjected to K-means clustering consisted of entropy and other metrics derived from the header fields. In contrast to the previous feature sets, when employing this feature set, the entire attack period was clustered at once instead of every 15 minutes, as done with the header fields and their combination. This decision was driven by the smaller data set size, considering that these measurements were calculated every minute instead of every two seconds. Examining the clustering results for router AR2 in Figure 6.44 allows the observation of the impact of this choice. Notably, the last four attacks appear to have been clustered into the "normal" cluster. This outcome is attributed to the Davies-Bouldin score for this clustering, which amounts to 0.56, slightly surpassing the predefined threshold. Consequently, the less compact cluster is defined as the attack cluster. However, the difference in compactness between the clusters is not substantial, and a high Davies-Bouldin score suggests suboptimal clustering. This suboptimal clustering could be attributed to the presence of numerous distinct attack types within the clustering period, posing challenges for the algorithm to correctly cluster the different attack vectors together. It might have been more appropriate to cluster this feature set in 15-minute intervals as well.

Increasing the window size of the calculations to 15 minutes causes the Davies-Bouldin score to fall slightly below the threshold, leading the cluster containing the last attacks to be identified as the attack cluster. Consequently, the first seven attacks are clustered as "normal", indicating that neither approach succeeds in clustering all attacks into the same cluster.

Analyzing the results for router AR1 reveals the presence of at least one alert for

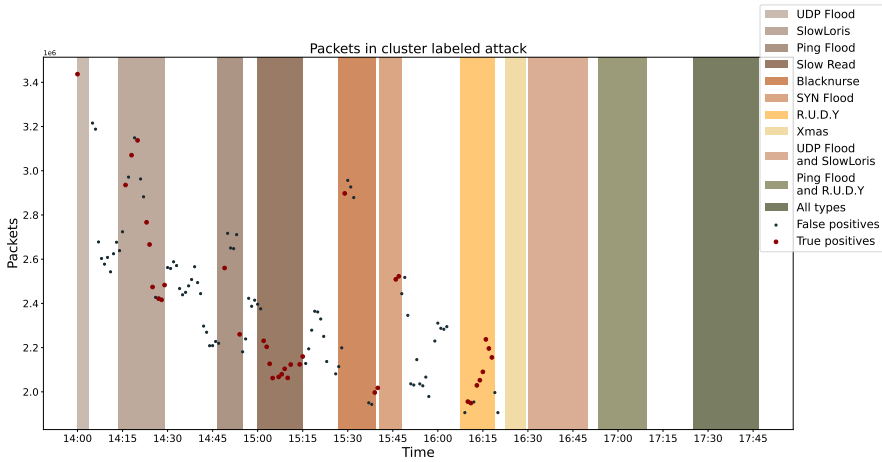


Figure 6.44: The number of packets in the attack cluster on attack generation round 3 using NetFlow entropy metrics as the feature set

every attack. Consequently, the clustering clusters at least some attack flows into the correct cluster. However, the non-attack cluster becomes significantly smaller, resulting in more false positives. The non-attack cluster grows as the window size increases, but the Davies-Bouldin score also rises, indicating increased difficulty in separating the clusters. Moreover, the clusters exhibit reduced compactness, potentially explaining the increase in the Davies-Bouldin score.

Similar trends are observed when applying the entropy feature set to the telemetry data on router AR2. Figure 6.45 illustrates the clustering results during the third round of attack generation. Here, it can be observed that the telemetry data set also manages to cluster a significant portion of the last three attacks into the incorrect cluster. The attacks predominantly clustered within the attack cluster include UDP flood, ping flood, Slow Read, and the Xmas attack.

In this case, the Davies-Bouldin scores for the three window sizes are notably high, with the 15-minute window size yielding a score of 0.97. This phenomenon leads to an increased similarity in cluster sizes, rendering it somewhat arbitrary whether the attacks are clustered together. The K-means algorithm may encounter challenges when clustering entropy metrics in general, as these metrics do not consistently exhibit the same behavior across different attack types. Once again, exploring the clustering at 15-minute intervals might have provided valuable insights into its potential effects.

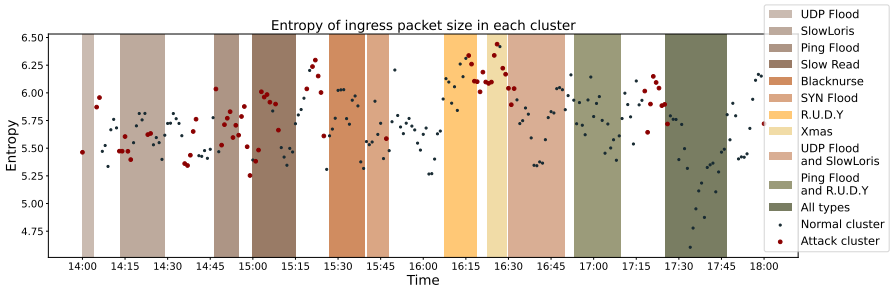


Figure 6.45: The entropy of ingress packet sizes in each cluster on attack generation round 3 using router interface telemetry entropy metrics as the feature set

Combined The K-means clustering performed on the combination of header fields and entropy metrics for router AR1 is presented in Figure 6.46. In this figure, it can be observed that there are true positives in the attack cluster for nearly all attacks. Attacks involving UDP floods and the SYN flood exhibit a higher number of true positives. However, attacks employing ping flood as one of their attack vectors demonstrate lower clustering accuracy. The UDP flood and SlowLoris attack yields the lowest Davies-Bouldin score, although all scores generally fall within a similar range. During the UDP flood attack, the cluster diameter of the attack cluster is over 12 times more compact than that of the non-attack cluster.

The Slow Read attack is one of the attacks that remain undetected. This attack has a Davies-Bouldin score below the threshold for massive attacks, but the attack flows in the Slow Read attack are less similar to each other compared to a flooding attack. Consequently, the cluster containing these attack flows is less compact than the cluster labeled as an attack. Since the Davies-Bouldin score is below the threshold, the algorithm erroneously assumes that the attack cluster should be compact, resulting in mislabeled clusters. If the threshold had been lowered to exclude this case from being classified as a massive attack, the algorithm would have mislabeled the SYN flood and Xmas attacks instead.

Considering the combined feature set on the router interface telemetry data for the same router (AR1), the clustering output can be observed in Figure 6.47. The K-means algorithm clusters most of the higher-volume measurements into the non-attack cluster and the remaining measurements into the attack cluster during most clustering periods. The highest Davies-Bouldin score is obtained when considering the combination of all attacks. This score may be elevated due to ongoing attacks throughout the entire clustering period, and the router interface telemetry data set is just counters on the interfaces that would get attack traffic at all times in this period.

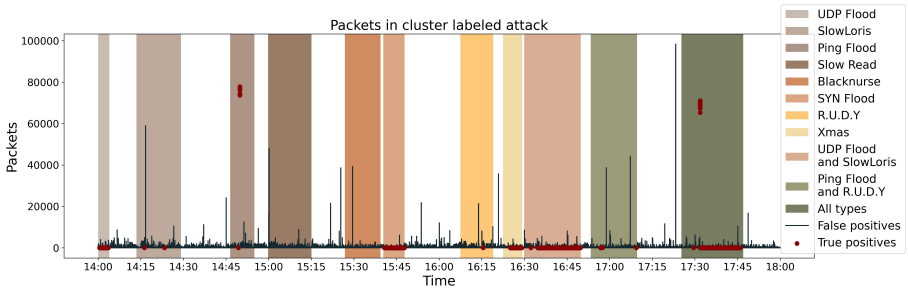


Figure 6.46: The number of packets in the attack cluster on attack generation round 3 using entropy metrics and header fields as the feature set

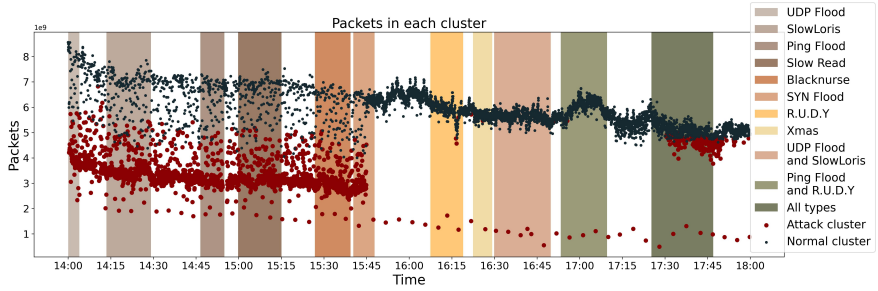


Figure 6.47: The number of packets in each cluster on attack generation round 3 using entropy metrics and router interface telemetry fields as the feature set

In theory, this situation would place all measurements in the attack cluster. However, the algorithm seems to encounter difficulties comprehending this scenario and adopts a different approach. Understandably, the K-means clustering does not perform optimally on the router interface telemetry data since it lacks as many distinguishing features as the NetFlow records.

6.4 Number of alerts

This section will present the number of alerts generated by the detection methods when they are combined. Table 6.5 displays the count of alerts produced by the detection methods during various tests. The tests used the IDS with three different sliding window sizes and two different methods for generating thresholds.

The table reveals that approximately 100 million alerts were generated during the four-hour duration of the third round of attack generation. This equates to an

average of 25 million alerts per hour, 420 000 alerts per minute, and 7,000 alerts per second. Over 90% of these alerts originate from the K-means detection methods. This is primarily due to the tendency of these methods to choose the largest cluster as the attack cluster, which may encompass millions of NetFlow records.

The ranking unit, responsible for assessing the perceived level of danger, receives an average of 672 combined alerts per minute during the same period. The significant difference in magnitude stems from the alert fusion process employed by the IDS, which combines multiple alerts based on their similarities.

	Detection methods	**Ranking**
TPR		
5 min	113 910 730	732
10 min	113 734 524	756
15 min	83 547 528	1 148
F1		
5 min	113 747 915	651
10 min	113 572 849	848
15 min	83 386 403	1 045

Table 6.5: Number of alerts generated by all detection methods combined and the number of combined alerts incorporated into the ranking

6.5 Performance

This section will present the accuracy of each detection method using the generated ground truth data. The aim is to address research question **R1**. Initially, an overview of the average precision for each detection method in the IDS and the alert fusion component will be provided based on the attack type. Subsequently, the performance of each detection method will be discussed in detail, utilizing various metrics defined in Section 2.3.1.

The average precision for each detection method is depicted in Figure 6.48. The figure illustrates the precision of all detection methods based on the ongoing attack type. The scores represent the average of tests conducted on the IDS. These tests encompassed three sliding window sizes: 5, 10, and 15 minutes. Additionally, two methods were employed to determine the thresholds, either by maximizing the TPR or the F1 score from the previous round of ground truth generation. When these

three sliding windows and two methods for threshold generation were combined, the result was six different tests. Each detection method was executed on the six routers containing ground truth data from the NetFlow dataset. The undefined cells in the figure correspond to methods that did not produce any alerts during the attack period.

It is worth noting that the UDP flooding attack exhibits numerous undefined cells due to the nature of the sliding window techniques utilized by the detection methods. These methods wait until the comparison window size has passed before comparing the current output to the mean of this window. Since the UDP flooding attack was relatively short, these methods had yet to stabilize before the attack concluded. Therefore it is advisable to start calculating the measurements for these methods before the attack. As a result of this issue, the UDP flooding attack was the *only* attack that went undetected and did not trigger any alerts from the ranking unit.

Examining the figure, it becomes apparent that several detection methods achieve a precision of 100% across all attacks. However, this may be misleading as the labeling of the telemetry dataset was not as accurate as the NetFlow dataset. In the telemetry dataset, every measurement within the attack period was labeled as an attack, thereby inflating the precision to 100%. The average precision across all detection methods for each attack is presented at the bottom of the figure. Notably, the Blacknurse attack appears to be the most challenging to generate accurate alerts for. Despite being a high-volume attack, it triggers more false than true alarms. However, this attack's average TPR remains relatively high, indicating many generated alerts. Understandably, the RF classifiers cannot detect this attack as it is unknown to the model. Contrarily, the SYN flood attack appears to be the easiest to generate accurate alerts for, as there are four methods specifically designed to detect SYN flows, and the attack was present during the training period of the RF classifiers.

When considering performance metrics other than precision, it is observed that the LRDDoS attacks exhibit higher FNR, with SlowLoris being the worst at 36%. Slow Read, on the other hand, demonstrates the best performance among the LRDDoS attacks. The elevated FNR implies that many events that should trigger an alert remain undetected during these attacks. Unfortunately, this outcome was expected, as detecting these attacks is challenging.

6.5.1 Empirical mean-variance model

The empirical mean-variance model had ten modules that generated deviation scores across 16 routers. Figure 6.49 displays the areas under the ROC curves for all ten modules on each of the 16 routers. Darker cells indicate higher AUC scores, while

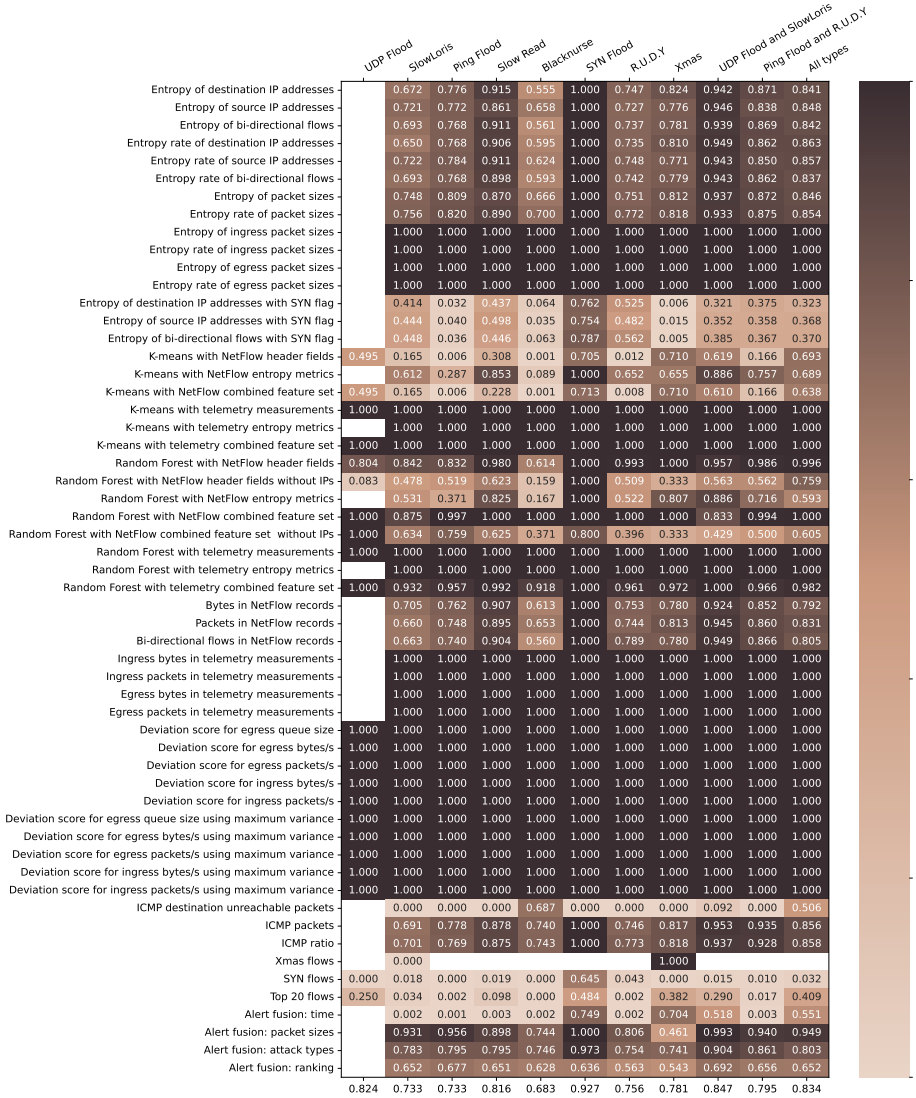


Figure 6.48: The precision on average of all detection methods in the IDS and the alert fusion

four cells are missing due to two routers lacking queue size measurements. The area under the ROC curve indicates the ease of setting a threshold that effectively separates true positives from false positives for a particular module and router.

The figure shows that distinguishing between true and false positives is generally easiest in the router closest to the victim. This router receives all the attack traffic, resulting in amplified attack patterns that are easier to differentiate. Additionally, features related to packets per second demonstrate slightly better performance, which is logical considering that the generated attacks primarily consist of numerous small packets.

Surprisingly, the maximum variance-based detection methods perform slightly better than the historical variance approach in most routers. This pattern also holds for the area under the precision-recall curve, although some routers exhibit higher FNR when the maximum variance is used. There are a few potential explanations for this observation. One possibility is that the models needed to be trained longer, preventing the variance from stabilizing. The training period lasted approximately two months, so perhaps in future work, this effect could be studied further. Another explanation could be that the weekly pattern is not as stable as anticipated, leading to inaccurate variances.

Overall, the model's performance is not exceptional, with most AUC scores around 0.5, indicating that it is random whether a threshold can effectively separate true positives from false positives. Two methods were tested for determining the appropriate threshold: maximizing the F1 score or the TPR from the previous attack. When using TPR to decide the threshold, the F1 scores for the routers and features were primarily around 0.7, while the TPR values ranged from 0.9 to 1.0. The FPR was consistently close to one for all routers. However, the maximum variance-based detection methods demonstrated lower FPR in certain routers when the threshold was based on TPR, albeit with a lower TPR as a consequence. On the other hand, when the threshold was determined based on the F1 score, the metrics exhibited more variation. F1 scores ranged from 0.7 to 0.8 on VR, and the TPR varied significantly, appearing worse for the maximum variance-based detection methods in certain routers.

6.5.2 Threshold: packets and bytes

For the detection method that employs a threshold on the number of packets or bytes in a time window, the threshold was determined like the empirical mean-variance model. The F1 score or TPR was maximized on each router using the ground truth data from the previous attack. To assess the general performance of the detection method, ROC curves were plotted for both detection methods. In Figure 6.50, dashed lines represent measurements on the ingress router interface telemetry data, while

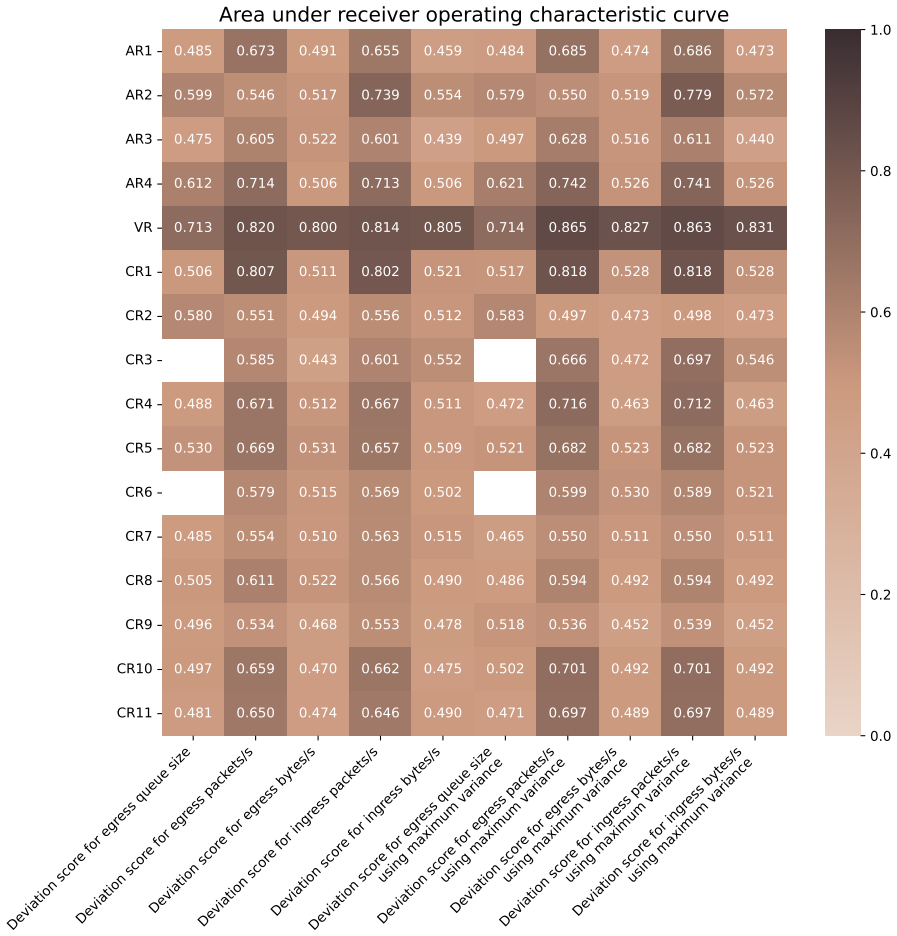


Figure 6.49: The area under the ROC curve for all features of the empirical mean-variance model on all routers

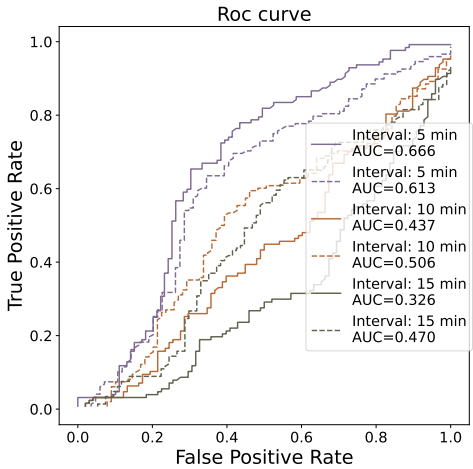


Figure 6.50: The ROC curve for all window sizes of the number of packets detection method

solid lines depict calculations from the corresponding NetFlow records on the same router. The figure illustrates the ROC curve for the number of packets on router AR1 during attack number three. It can be observed that the AUC score is highest when using the smallest window size. However, the AUC score is not particularly high, indicating limited promise for this detection method.

Multiple factors influence the performance of a detection method, so various parameters were varied to assess their effects. The window size, sliding window movement frequency, and comparison window size were varied, and the top 10 AUC scores for the number of packets on router CR3 using NetFlow records are presented in Table 6.6. The AUC score obtained with the initially chosen variables would rank at best 371st. However, the differences in scores are minimal, as evident from the table. Since the variations in the scores are small, the specific choice of variables may not have a significant impact, as none exceeds 0.652. This observation might be attributed to the router used for parameter variation, which could be explored further in future research. It might also suggest that this is not the most effective detection method, but it still outperforms the random separation of data into true and false positives.

The same was done on the router interface telemetry data set on VR. One of the best combinations across the features was a frequency of 1 second, a window size of 5 seconds, and a comparison window size of 8 seconds. When this combination was used, the area under the ROC curve was increased by an average of 45% across the features. The areas under the precision-recall curves were even higher than the

Frequency	Window Size	Comparison Window Size	Value
10 s	13 min	12 min	0.652
10 s	13 min	13 min	0.652
10 s	13 min	11 min	0.652
50 s	8 min	8 min	0.651
10 s	13 min	14 min	0.651
10 s	13 min	10 min	0.651
50 s	8 min	7 min	0.650
10 s	13 min	15 min	0.650
10 s	13 min	16 min	0.649
50 s	9 min	7 min	0.649

Table 6.6: AUC for ROC curve using different variables for the number of packets in NetFlow records

ROC curves. The best performance is achieved when the measurements are done frequently and with a small window size. This indicates that using a sliding window technique might not have been beneficial for detecting attacks on the telemetry data. It might have been better to rather have a threshold directly on the measurements.

The impact of the two threshold selection methods can be observed in Figure 6.51. The figure displays the F1 scores of different detection methods depending on the chosen metric for threshold maximization. The scores exhibit considerable variation, and it appears that the number of packets in a 10-minute window using the router interface telemetry dataset achieves the best scores across all routers when F1 is used for threshold decision. When maximizing TPR, most F1 scores are around 0.7, but router CR3 using NetFlow records demonstrates nearly identical scores. While it may seem that NetFlow performs worse overall, this observation could be attributed to labeling all measurements during attack periods as positives, which likely impacted the performance.

6.5.3 Threshold: SYN flag

As discussed in Section 6.3.3, the hypothesis regarding an increase in the number of packets in SYN flood flows did not appear to be accurate. This observation is supported by the ROC curve in Figure 6.52, where the AUC for all routers is either around 0.5 or below. This suggests that differentiating between true and false positives is random. Router AR2 exhibits the poorest performance, possibly due to its connection to an IXP and the high volume of external traffic it receives, some of which may contain undetected attacks and result in mislabeling them as

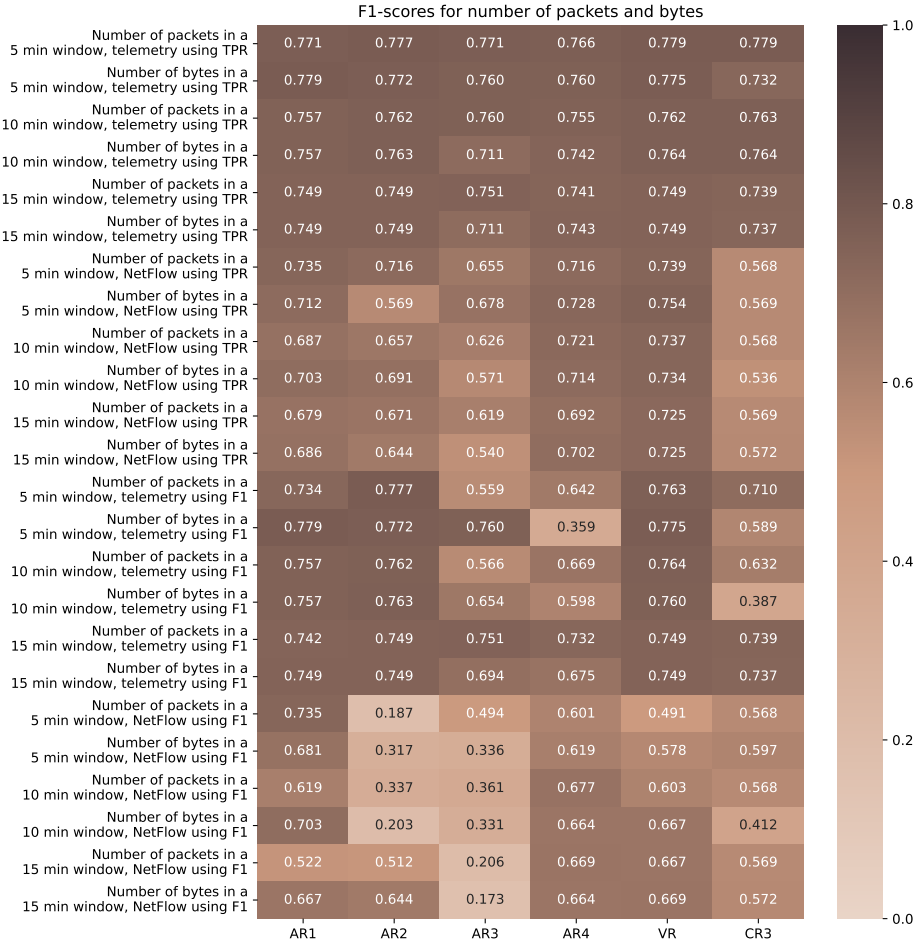


Figure 6.51: The F1 score for all window sizes of the number of packets and bytes detection methods when maximizing F1 in the threshold decision

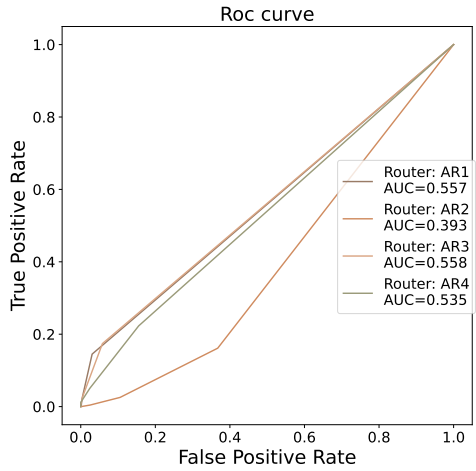


Figure 6.52: The ROC curve for all routers of the number of packets in an SYN flow detection method during the third round of attack generation

benign. However, this behavior is not consistent across the three generated attacks. For the first two attacks, router AR4 displayed the worst performance. None of the attack flows in this router had more than two packets, making it challenging to establish an effective threshold. In both AR3 and AR4, the false positives would have been eliminated if the threshold were set to four packets per flow. However, such a threshold would lose 99.5% and 98.8% of the true positives, respectively.

6.5.4 Threshold: URG, PSH, and FIN flag

Table 6.7 illustrates the precision of the check performed on flows with the URG, PSH, and FIN flags set. AR1, AR3, and AR4 exhibit 100% precision, indicating that all alerts were true positives. As mentioned in Section 6.3.4, CR2 also includes additional flows that might pertain to another Xmas attack, potentially resulting in higher precision for this router. On the other hand, routers VR and CR3 do not detect any flows with all the flags mentioned above set to one since they only capture flows from the victim.

6.5.5 Threshold: ICMP destination unreachable

In Figure 6.53, the accuracy of the threshold set on ICMP destination unreachable packets is not ideal in the first three attack routers. However, if the threshold is chosen based on the F1 score, the accuracy appears slightly improved. Router AR4 and the victim router exhibit relatively decent accuracy, which is promising. One possible explanation for the poor performance in AR1, AR2, and AR3 is the absence

Router	Precision
AR1	1.00
AR2	0.98
AR3	1.00
AR4	1.00
VR	Undef
CR3	Undef

Table 6.7: Precision on detecting Xmas attack flows for all routers

of anomalous ICMP destination unreachable packets during attack number two, as they are primarily associated with the Blacknurse attack. Consequently, when selecting the threshold for detecting subsequent attacks, AR1, AR2, and AR3 had insufficient data to rely on from the second round of attack generation. In such cases, the algorithm calculates the threshold based on the mean of the data from other routers. On the other hand, despite also lacking attack flows from round two, AR4 seemed to benefit from using the mean threshold. This observation might be attributed to AR4 having the lowest traffic volume, allowing for a lower threshold setting.

Figure 6.54 displays the ROC curve generated using the ground truth from attack number three. It suggests that choosing the threshold based on this attack could have significantly improved the results. Interestingly, VR and CR3 appear to struggle in differentiating between true and false positives, deviating from what was observed in Figure 6.53. This contrast can be attributed to the threshold being set quite low, and the accuracy metric does not consider the precision of the alerts, which was nearly zero for all detection methods on VR and CR3. This emphasizes the importance of selecting an appropriate threshold, as it can greatly impact the overall performance.

6.5.6 Threshold: ICMP packets and ICMP ratio

In Figure 6.55, the FNR for all routers and combinations of detection methods using ICMP packets and ratio is depicted. It can be observed that when the threshold is chosen based on the maximum TPR from the previous attack, the TNR is significantly lower compared to using F1 scores. This difference arises because the thresholds set using ground truth data from the second round of attack generation are much higher than those derived from the third round when maximizing the F1 score. However, during the third round of attack generation, precision is marginally better when employing the F1 score-based threshold, which can be advantageous in reducing false alerts.

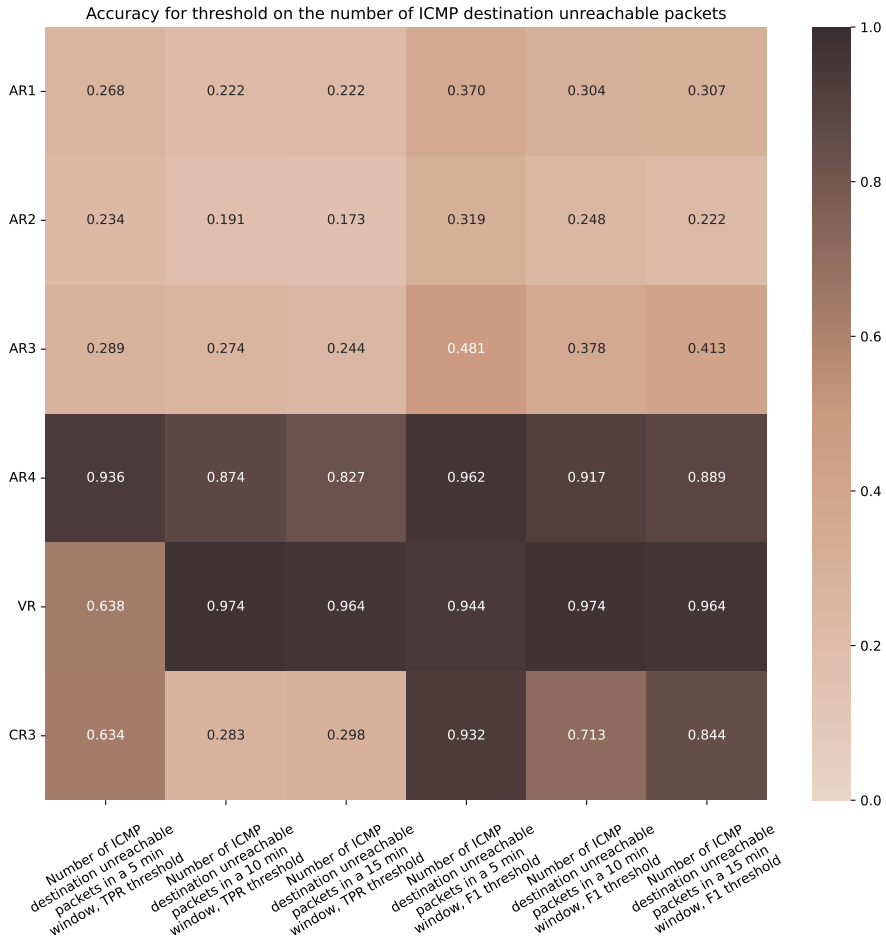


Figure 6.53: The accuracy for all routers of the number of ICMP destination unreachable packets during the third round of attack generation

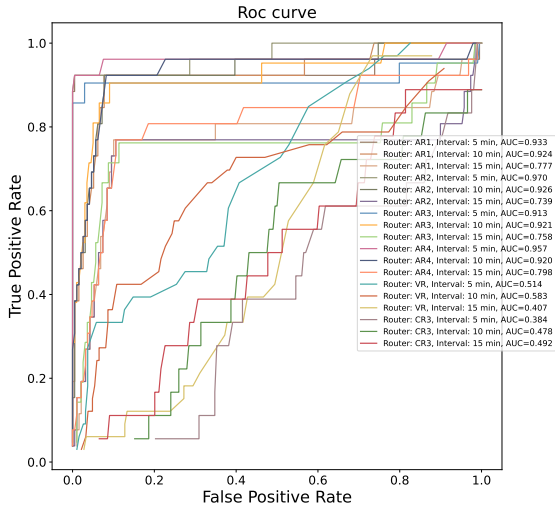


Figure 6.54: The ROC curve for all routers of the number of ICMP destination unreachable packets during the third round of attack generation

Furthermore, the impact of different variable choices on performance was explored. Table 6.8 presents the top 10 combinations of variables for ICMP packets. The results indicate that an optimal frequency for moving the sliding window would be every five seconds, with a window size of approximately 13 minutes. The comparison window size exhibits more variability, although many of the top values fall within the 70-80 seconds range. While the highest value slightly surpasses the best value for CR3 using the original variable combination, the difference is not substantial. This could arise from labeling all measurements with an attack flow in the records for the current frequency as an attack. Consequently, the labeled attack may not involve ICMP packets, thereby distorting the performance of these metrics in detecting ICMP-related attacks.

For the ICMP ratio, the ideal frequency appears to be every 10 seconds, with a window size of around 16 minutes. The comparison window size demonstrates slightly more variation, with the top five values exceeding 150 seconds.

6.5.7 Threshold: bi-directional flows

The accuracy of the detection method utilizing a threshold on the number of bi-directional flows is presented in Figure 6.56. It can be observed from the heatmap that, regardless of the threshold decision method employed, the accuracy remains

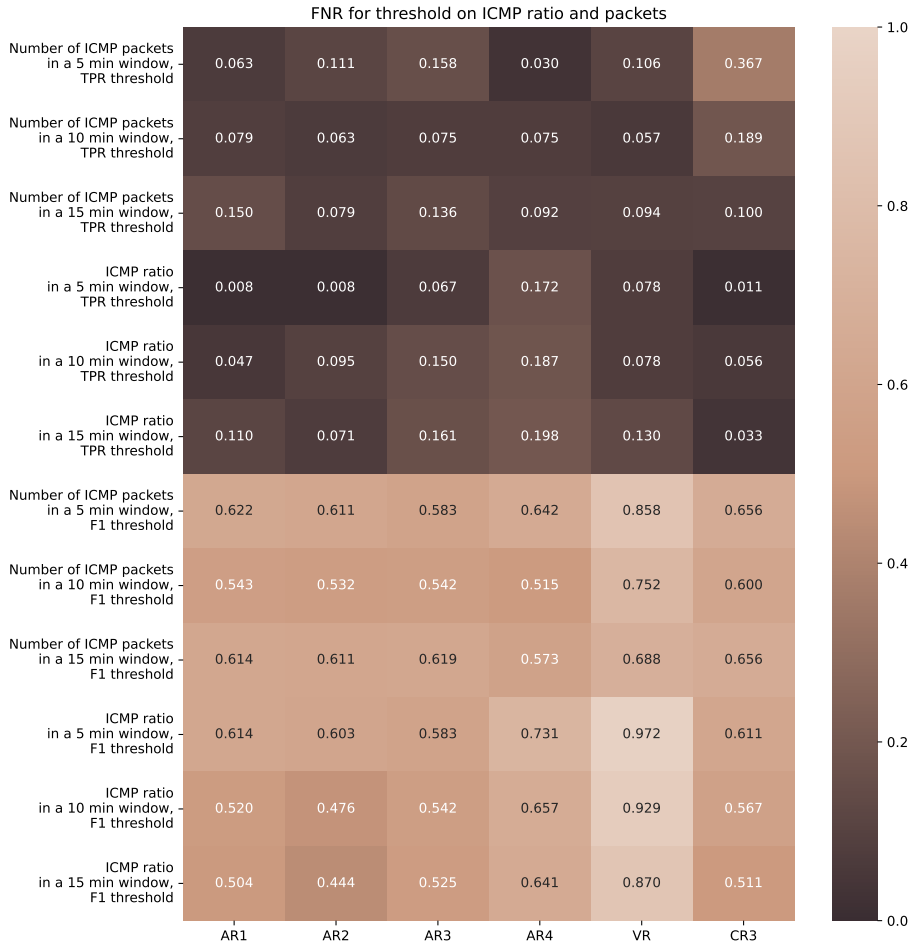


Figure 6.55: The FNR for all routers of the ratio and number of ICMP packets during the third round of attack generation

Frequency	Window Size	Comparison Window Size	AUC
5 s	12 min	10 s	0.780
5 s	12 min	15 s	0.779
5 s	13 min	15 s	0.778
5 s	14 min	75 s	0.778
5 s	14 min	70 s	0.777
5 s	14 min	80 s	0.777
5 s	14 min	85 s	0.776
5 s	14 min	90 s	0.776
5 s	14 min	65 s	0.776
5 s	13 min	10 s	0.776

Table 6.8: AUC for ROC curve of ICMP packets in router CR3 during attack generation round 3

approximately 0.5. This pattern is also evident in the area under the ROC curves for the third round of attack generation. These findings suggest the difficulty in finding a threshold that effectively distinguishes true positives from false positives. Various combinations of variables were calculated to explore whether the chosen variables influenced the performance. This yielded AUC values of up to 0.6 when the frequency was set to 10 seconds, the window size was approximately 17 minutes, and the comparison window spanned around 200 seconds. Although the increase is not substantial, it could potentially be higher if further combinations of parameters were explored.

6.5.8 Entropy and entropy rate: packet size

The performance of the entropy and entropy rate of packet sizes was evaluated on both NetFlow and router interface telemetry data sets. Figure 6.57 presents the F1 scores for all combinations of detection methods applied to packet sizes. The highest scores, around 0.75, were observed, with better performance observed on the telemetry data set. This was further amplified when using the F1 score as the basis for the threshold decision. However, using this threshold, the less heavily trafficked routers (AR4, VR, and CR3) exhibited poorer performance. However, when using the TPR as the basis for thresholds, the victim router did not demonstrate the worst performance. Generally, the entropy rate outperformed entropy, aligning with the findings of [21]. Smaller window sizes also tended to yield slightly better performance. An exploration of different parameters was conducted to investigate potential performance improvements. While the entropy and entropy rate of packet sizes in the NetFlow records showed a minor improvement of 0.06, it was not deemed

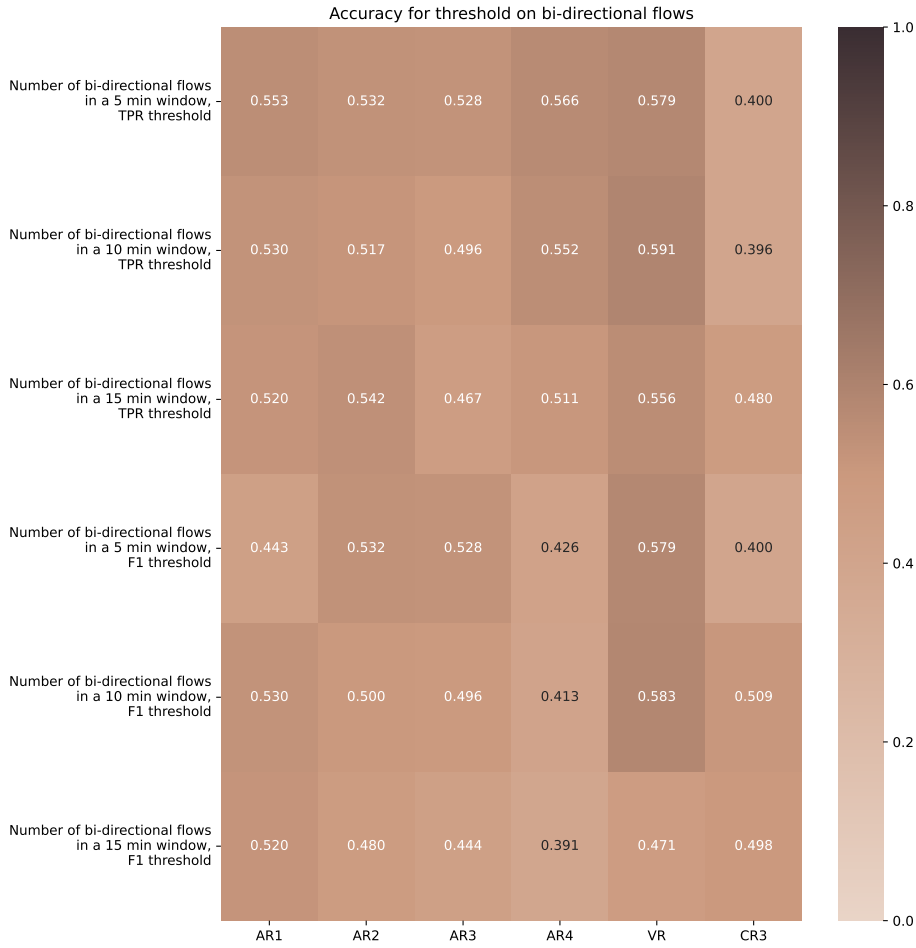


Figure 6.56: The accuracy for all routers of the number of bi-directional flows during the third round of attack generation

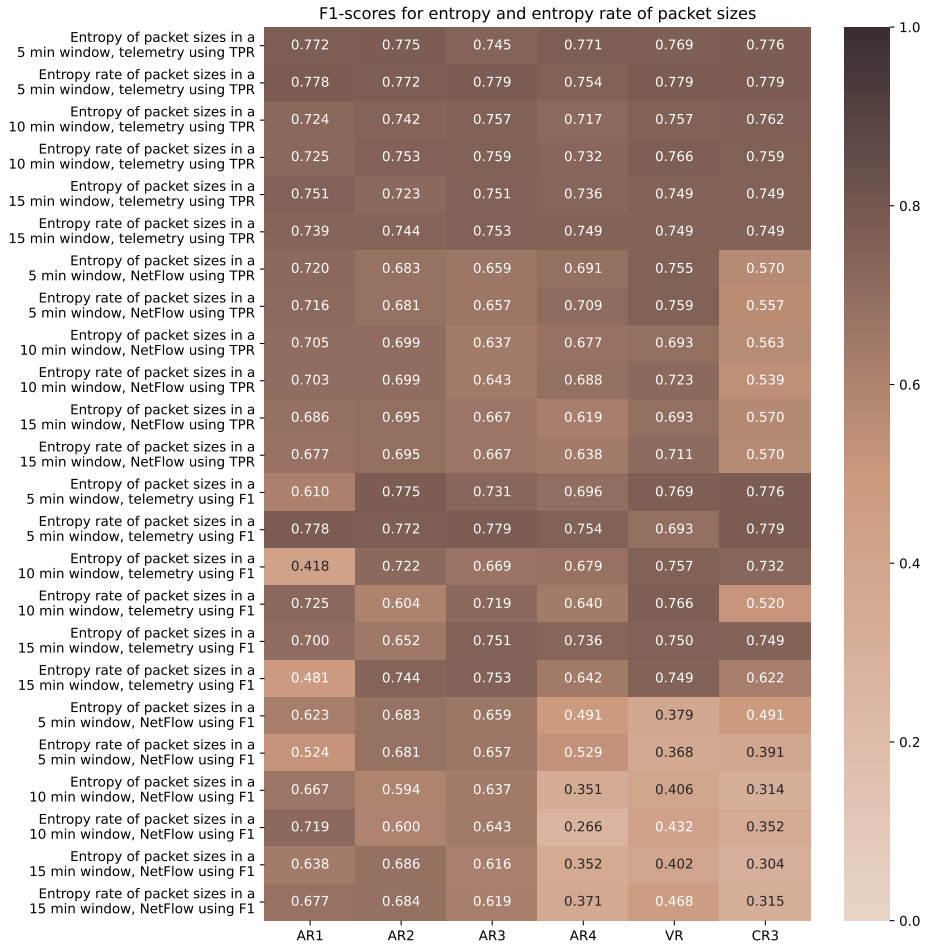


Figure 6.57: The F1 scores for all routers of the entropy and entropy rate of packet sizes during the third round of attack generation

Frequency	Window Size	Comparison Window Size	α	AUC
60 s	1 min	20 min	12	0.782
60 s	1 min	20 min	10	0.782
60 s	1 min	20 min	14	0.782
60 s	1 min	20 min	15	0.781
60 s	1 min	20 min	13	0.781
60 s	1 min	20 min	11	0.781
60 s	1 min	20 min	8	0.781
60 s	1 min	20 min	9	0.781
60 s	1 min	20 min	7	0.779
50 s	2 min	17 min	15	0.778

Table 6.9: AUC for the ROC for entropy of packet sizes in NetFlow records

significant enough to make a substantial difference. However, the AUC score for the precision-recall curve improved by approximately 0.17, which could potentially have a more noticeable impact. The optimal parameter choices included a frequency of 1 minute, a window size of 3 minutes, and a comparison window size of 20 minutes. The entropy rate consistently displayed these parameter values among its top 9 combinations for the area under the ROC curve, as presented in Table 6.9. The parameter α appeared to have the least impact on performance, aligning with the AUC score for the precision-recall curve. It is important to note that all the top 9 combinations involve the endpoints of the explored parameters. Hence, it is possible that better performance could be achieved with smaller window sizes or larger comparison window sizes. However, given that the best AUC score for the precision-recall curve was associated with a window size of 3 minutes, it might just be the comparison window that could have increased the performance. It is also possible that this method is unsuitable for detecting DDoS attacks.

The same exploration on the telemetry data set showed the most significant improvement in the entropy rate of packet sizes. The highest scoring combination was a frequency of almost 3 minutes, an interval of less than a minute, a comparison window of around 20 minutes, and an α of 3. The AUC was improved by at most 37.6%, giving a value of 0.841. From exploring different combinations, the variable that influences the score the most is the window size. Smaller window sizes yielded the worst AUC for the ROC curve.

Figure 6.58: The F1 scores for all routers of the entropy and entropy rate of destination IP addresses during the third round of attack generation

6.5.9 Entropy and entropy rate: destination IP addresses

Similar to other detection methods, thresholds on the change of entropy and entropy rate for destination IP addresses were utilized, employing either the TPR or F1 score as the threshold basis. The F1 scores during the third round of attack generation, shown in Figure 6.58, do not align with the expectation that choosing the threshold based on the F1 score would yield higher scores. Precision improves for specific routers with F1-based thresholds, but the TPR decreases due to the higher threshold, resulting in slightly lower F1 scores. Notably, the lowest score is observed for entropy in a 15-minute window on AR1 when using the F1 score as the basis for the threshold. This router exhibits poor precision and TPR, partly due to the higher threshold derived from ground truth data from the second round of attack generation compared to the third. Furthermore, this combination also yields the lowest AUC for the ROC curve during round 3, indicating the difficulty in finding an appropriate threshold to distinguish true from false positives.

Similar to the entropy and entropy rate of packet sizes, slight performance improvements could be achieved by using a smaller window size and a larger comparison

window. The entropy rate would also perform better with a higher value of α. This may be attributed to a higher window size providing a more stable entropy value, potentially allowing minor attacks to go undetected. Interestingly, the worst performance for the entropy rate occurs when using an α value of two, which is very close to Shannon's entropy. Despite its popularity, this choice yields the poorest performance.

6.5.10 Entropy and entropy rate: source IP addresses

The F1 scores for the entropy and entropy rate of source IP addresses are displayed in Figure 6.59, showing overall better performance compared to the destination IP addresses. Utilizing the threshold based on the F1 scores yields particularly improved results. This could be attributed to the similarity of thresholds between attacks two and three for the source IP addresses, unlike the entropy and entropy rate of the destination IP addresses. Additionally, the greater variability in the source entropy rate might facilitate easier differentiation of attack traffic. The areas under the ROC curves for the source IP addresses are slightly higher, suggesting better performance.

Similarly, employing a lower window size and a larger comparison window would have led to slight performance improvements. It is worth noting that the entropy of source IP addresses performed best with an α value of around 5, differing from other entropy metrics. On the other hand, the entropy rate demonstrated better performance with a higher α value. However, considering the limited gain in performance, other factors may contribute to the limited effectiveness of entropy as a detection method.

6.5.11 Entropy and entropy rate: bi-directional flow

The F1 scores of the entropy and entropy rate of bi-directional flows exhibit similarities to those of the destination IP addresses but with slightly greater stability. Overall, the entropy and entropy rate of bi-directional flows demonstrate better F1 scores, TPR, and FNR than both source and destination IP addresses, but they have the worst precision, accuracy, and FPR. This suggests that the threshold for bi-directional flows is lower, resulting in more generated alerts. However, the effectiveness of this threshold depends on the ground truth data from the second round of attack generation.

Analyzing the area under the ROC and precision-recall curves for the three entropy metrics during the third round of attack generation reveals that the entropy and entropy rate of bi-directional flows do not achieve the best performance on average. The entropy and entropy rate of source IP addresses consistently yield higher values for both curves. It is important to note that even the highest average

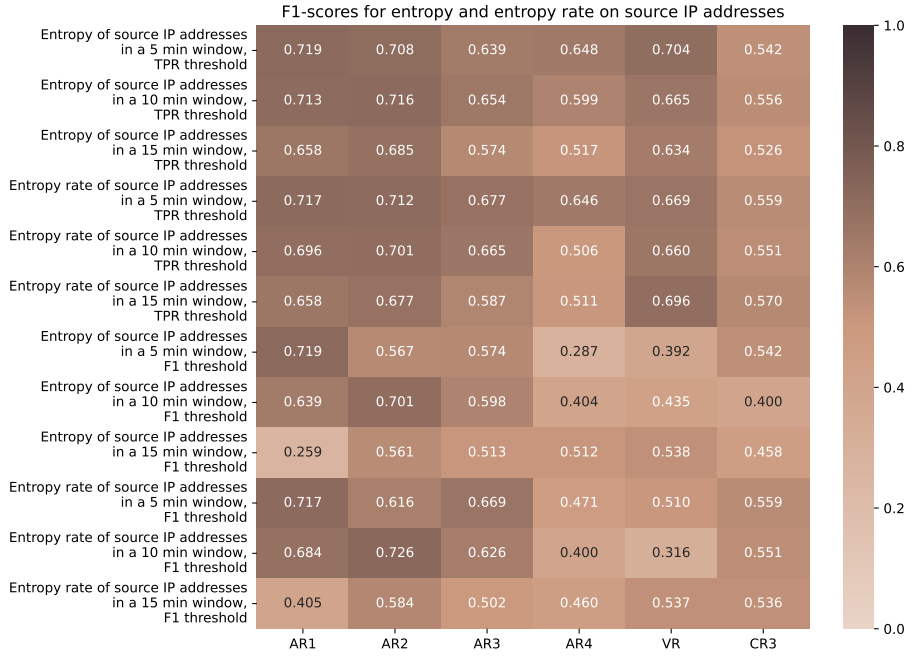


Figure 6.59: The F1 scores for all routers of the entropy and entropy rate of source IP addresses during the third round of attack generation

value is only 0.55, indicating poor performance overall. Interestingly, the entropy rate of bi-directional flows performs better with larger window sizes compared to source IP addresses in specific routers. This behavior may be attributed to the more stable nature of the entropy rate in bi-directional flows, which might make attacks stand out more.

For the entropy of bi-directional flows, slight performance improvements can be achieved using a lower window size and a larger comparison window, while the entropy rate benefits from larger window and comparison window sizes. Regarding the entropy rate of destination IP addresses and the entropy and entropy rate of bi-directional flows, both metrics demonstrate improved performance with lower values of α. The optimal performance for both entropy metrics occurs when α is set to three. This observation could suggest that sudden significant changes are more noticeable in larger windows. Notably, a trade-off between lower α and larger window size is observed when examining the top AUC values for the precision-recall curve of the entropy rate. This indicates that achieving good performance in the entropy rate requires contrasting values of α and window size. Higher alpha values increase

entropy variations, while larger window sizes promote stability. Consequently, a balance must be struck when deciding the threshold to differentiate between true and false positives.

6.5.12 Entropy: SYN flows

The performance of the entropy of SYN flows during the third round of attack generation could be better. The F1 scores for this metric can be observed in Figure 6.60. When using the F1 score as the basis for the threshold, most F1 scores for the victim router are undefined. This is due to the absence of true positives for VR and CR3 throughout the attacks. Since these routers only capture flows from the victim to the attackers, which do not exclusively contain the SYN flag, they do not encounter anomalous SYN flows. The defined cells for these two routers exist because false positives occur during the attack period.

Notably, the entropy of destination IP addresses in a 15-minute window, using the TPR-based threshold, yields a FPR exceeding 90% for these two routers. None of the F1 scores present promising results, which may be attributed to the threshold choices. This observation is further confirmed by the significantly higher area under the ROC curve compared to the precision-recall curve. This contrast suggests that by setting the threshold high enough, false positives can be reduced, but at the expense of increased false negatives. Notably, the area under the precision-recall curve is the smallest for the source IP addresses, which contradicts the findings for all the flows. However, the area under the ROC curve is not the worst for source IP addresses, but bi-directional flows perform better.

6.5.13 Top 20 destination IP address flows

The detection method applied to the top 20 destination IP addresses did not yield satisfactory results either. Table 6.10 presents the performance metrics for this detection method. Interestingly, the same values were obtained when using both threshold decision methods. This can be attributed to the fact that most attack flows entered the distribution as the top flow, resulting in a threshold of 0.99. Upon examining the FNR, it is evident that most attack flows do not appear among the top 20 flows, thus not triggering an alert. If an alert is triggered, the maximum probability of it being a true positive is 19.8%. The accuracies generally exhibit acceptable values because many true negatives are correctly classified. However, it is worth noting that this is likely due to the flows not being substantial enough to be classified as attack flows by this method.

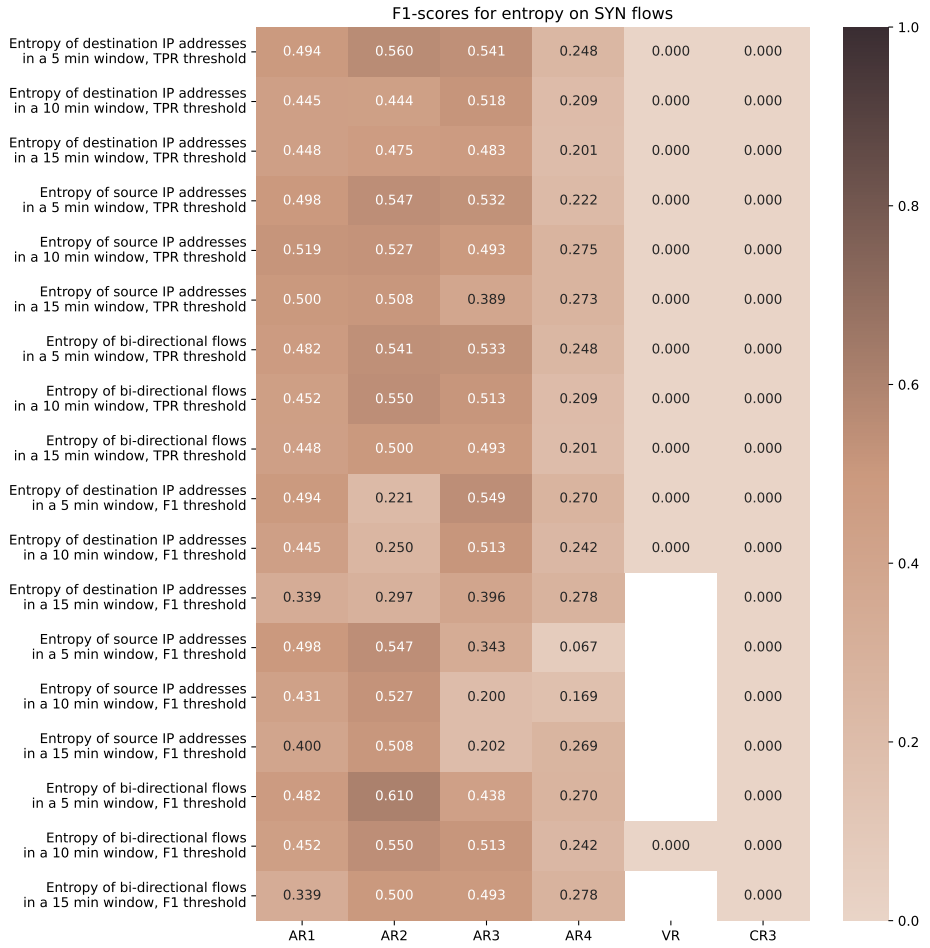


Figure 6.60: The F1 scores for all routers of the entropy of SYN flows during the third round of attack generation

Router	F1	Precision	TPR	Accuracy	FPR	FNR
VR	$4.8 \cdot 10^{-4}$	0.197	$2.4 \cdot 10^{-4}$	0.011	0.083	1.000
AR3	$3.0 \cdot 10^{-4}$	0.128	$1.5 \cdot 10^{-4}$	0.602	$6.8 \cdot 10^{-4}$	1.000
CR3	$3.0 \cdot 10^{-4}$	0.052	$1.5 \cdot 10^{-4}$	0.907	$2.8 \cdot 10^{-4}$	1.000
AR1	$1.9 \cdot 10^{-4}$	0.089	$1.0 \cdot 10^{-4}$	0.841	$1.9 \cdot 10^{-4}$	1.000
AR4	$1.8 \cdot 10^{-4}$	0.198	$9.1 \cdot 10^{-5}$	$5.0 \cdot 10^{-3}$	0.070	1.000
AR2	$1.4 \cdot 10^{-4}$	0.078	$7.3 \cdot 10^{-5}$	0.855	$1.4 \cdot 10^{-4}$	1.000

Table 6.10: Performance for top 20 destination IP addresses

6.5.14 Random Forest

A RF classifier was trained on five different feature sets using NetFlow records from all 16 routers. The classifier's performance was evaluated, and it was observed that CR3 exhibited the highest performance across almost all feature sets. This observation could be attributed to two possible reasons. Firstly, CR3 receives a relatively lower amount of attack traffic, as it only processes the traffic from the victim back to attacker 3. Secondly, the traffic passing through CR3 may be more similar to the training data since certain unknown attacks, such as Blacknurse and UDP flood, do not necessitate a response from the victim.

Fields The F1 score of CR3 on this specific feature set (Table 6.11) was 99.8%, indicating nearly perfect performance. However, the other routers yielded less promising results on this feature set. Due to the lack of alerts generated by these routers, a substantial portion of the attack flows went undetected. Among the other routers, VR exhibited the best performance, which could be interesting to detect as the effects of attacks on a victim also indicate an attack.

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.998	1.000	0.997	1.000	$1.1 \cdot 10^{-5}$	$3.0 \cdot 10^{-3}$
VR	0.455	1.000	0.295	0.303	$5.0 \cdot 10^{-4}$	0.705
AR4	0.339	1.000	0.204	0.209	$2.3 \cdot 10^{-4}$	0.796
AR1	0.325	1.000	0.194	0.872	$4.3 \cdot 10^{-6}$	0.806
AR3	0.316	1.000	0.188	0.677	$2.4 \cdot 10^{-6}$	0.812
AR2	0.213	1.000	0.119	0.872	$4.5 \cdot 10^{-7}$	0.881

Table 6.11: Random Forest performance when using NetFlow header fields as the feature set

Upon removal of IP addresses from the feature set, a significant decline in performance was observed specifically in CR3, suggesting that its high performance was primarily based on learning the malicious nature of the IP addresses employed in the training data (see Table 6.12). However, the performance on the victim router does not decrease much, but it produces slightly more false positives. The presence of more false positives without IP addresses is logical since the model's confidence in detecting attack patterns decreases. Notably, the accuracy appears to be high for both feature sets in AR1, AR2, and AR3 due to the scarcity of alerts generated by these routers, leading to most negatives being true negatives. This highlights the limitation of accuracy as a performance metric when there is an imbalance between the number of positive and negative instances. This limitation is further exemplified by the lower accuracy achieved by AR4 and VR, which have a higher number of positives than negatives. AR3 has the most similar number of positives and negatives, which would give the most reliable accuracy score.

Router	**F1**	**Precision**	**TPR**	**Accuracy**	**FPR**	**FNR**
CR3	0.459	0.999	0.298	0.935	$3.2 \cdot 10^{-5}$	0.702
VR	0.454	1.000	0.294	0.302	$8.7 \cdot 10^{-4}$	0.706
AR4	0.303	1.000	0.178	0.183	$6.4 \cdot 10^{-3}$	0.822
AR2	0.207	0.985	0.116	0.872	$3.0 \cdot 10^{-4}$	0.884
AR3	0.200	0.999	0.111	0.646	$5.9 \cdot 10^{-5}$	0.889
AR1	0.199	0.997	0.110	0.858	$5.7 \cdot 10^{-5}$	0.890

Table 6.12: Random Forest performance when using NetFlow header fields without IP addresses as the feature set

The performance of the classifier trained on router interface telemetry data was assessed for all routers, and the results are displayed in Table 6.13. The victim router consistently outperformed the other routers across various metrics. However, the precision was slightly affected by a relatively higher FPR. The improved performance of VR from the NetFlow records can be attributed to its inclusion of traffic destined for the victim, in addition to the feedback traffic from the victim. Furthermore, CR1, the router situated two hops away from the victim, demonstrated decent performance, indicating its potential for early detection of attacks within the network. Notably, CR5 exhibited the highest precision, as it generated fewer highly accurate alerts. These alerts can be valuable in identifying attacks at an earlier stage in the network.

Entropy The performance of the RF classifier was evaluated using entropy feature sets. It was observed that the overall performance could have been significantly improved. However, CR3 consistently demonstrated the best performance across

Router	F1	Precision	TPR	Accuracy	FPR	FNR
VR	0.809	0.855	0.768	0.789	0.182	0.232
CR1	0.760	0.909	0.652	0.759	0.092	0.348
CR3	0.666	0.599	0.748	0.561	0.702	0.252
AR2	0.647	0.657	0.637	0.594	0.466	0.363
AR3	0.530	0.625	0.460	0.524	0.387	0.540
CR10	0.446	0.633	0.344	0.500	0.279	0.656
CR11	0.374	0.571	0.278	0.456	0.293	0.722
CR6	0.315	0.522	0.226	0.427	0.290	0.774
AR4	0.302	0.684	0.194	0.477	0.126	0.806
CR5	0.279	0.911	0.165	0.503	0.022	0.835
CR7	0.242	0.589	0.152	0.443	0.150	0.848
CR9	0.226	0.596	0.139	0.443	0.132	0.861
AR1	0.141	0.530	0.082	0.422	0.101	0.918
CR2	0.116	0.565	0.065	0.425	0.070	0.935
CR8	0.110	0.519	0.061	0.419	0.080	0.939
CR4	0.069	0.816	0.036	0.432	0.011	0.964

Table 6.13: Random Forest performance when using router interface telemetry measurements as the feature set

all window sizes, except for the 5-minute window (see Table 6.14). The 5-minute window exhibited a notable increase in false negatives and false positives, resulting in significantly poorer precision than the header fields feature sets. This could suggest that the model encounters greater difficulty in identifying patterns in the entropy metrics compared to the header fields. The accuracy appeared to be highest in CR3, which can be attributed to its lower number of alerts compared to VR and AR4 and a smaller number of positive instances, resulting in a higher count of true negatives.

With the use of a 10-minute window, CR3 demonstrated far better TPR and FNR as depicted in Table 6.15. This improvement is likely due to a higher number of alerts generated by CR3, leading to alerts being triggered on most of the traffic, resulting in correct detections at times. However, this increase in alerts led to reduced accuracy and precision. Notably, AR1 ceased to produce alerts altogether. Despite its poor performance in the 5-minute window, this complete absence of alerts suggests that AR1 and AR2, being highly active routers, pose challenges for the RF algorithm to distinguish between attack and benign traffic. Additionally, mislabeled unknown attacks in the training data might contribute to this reduction in performance.

The 15-minute window exhibited slightly improved performance in some routers,

Router	F1	Precision	TPR	Accuracy	FPR	FNR
VR	0.329	0.833	0.205	0.566	0.044	0.795
AR4	0.286	0.633	0.184	0.596	0.083	0.816
CR3	0.265	0.405	0.197	0.647	0.138	0.803
AR3	0.202	0.361	0.140	0.562	0.162	0.860
AR1	0.065	0.129	0.043	0.511	0.189	0.957
AR2	0	undef	0	0.600	0	1

Table 6.14: Random Forest performance when using entropy as the feature set for a 5 minute window

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.453	0.308	0.855	0.317	0.948	0.145
AR4	0.408	0.593	0.311	0.596	0.173	0.689
VR	0.314	0.540	0.221	0.487	0.213	0.779
AR3	0.195	0.325	0.140	0.535	0.197	0.860
AR1	0	undef	0	0.600	0	1
AR2	0	undef	0	0.591	0	1

Table 6.15: Random Forest performance when using entropy as the feature set for a 10 minute window

as shown in Table 6.16. The green cell in the table denotes the highest F1 score among all window sizes. The enhanced performance can be attributed to an increased number of alerts, evident from the rise in FPR and TPR in VR and AR4, as well as a decrease in FNR for these routers. AR1 resumed generating alerts; however, all alerts were false positives.

If precision alone is emphasized, the 5-minute window would be the preferred choice, albeit at the cost of missing numerous attacks. On the other hand, if maximizing the detection of attacks (higher TPR) is prioritized, the 15-minute window yields slightly better TPR on average for all routers. The 10-minute window exhibited the lowest FNR on average; however, the FNR for the 15-minute window was only slightly worse. The overall performance suggests that the 15-minute window may be more suitable for detecting more attacks, while the 5-minute window strikes a balance between accuracy and attack detection.

Regarding the entropy metrics on the router interface telemetry data set, the performance during the 5-minute window is depicted in Table 6.17. VR displayed the best performance among the routers in this feature set, followed by CR1. However,

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.491	0.379	0.697	0.511	0.584	0.303
AR4	0.443	0.520	0.386	0.564	0.290	0.614
VR	0.433	0.568	0.350	0.511	0.305	0.650
AR3	0.141	0.250	0.098	0.511	0.203	0.902
AR1	0	0	0	0.564	0.045	1
AR2	0	undef	0	0.582	0	1

Table 6.16: Random Forest performance when using entropy as the feature set for a 15 minute window

CR1 exhibited better precision than VR, indicating fewer but more accurate alerts. None of the F1 scores in this feature set were particularly remarkable, consistent with the findings in the NetFlow data set.

Router	F1	Precision	TPR	Accuracy	FPR	FNR
VR	0.682	0.788	0.601	0.654	0.261	0.399
CR1	0.616	0.820	0.493	0.621	0.174	0.507
CR2	0.558	0.655	0.486	0.525	0.413	0.514
AR3	0.510	0.595	0.446	0.471	0.489	0.554
AR4	0.466	0.643	0.365	0.483	0.326	0.635
CR11	0.453	0.616	0.358	0.467	0.359	0.642
AR2	0.430	0.697	0.311	0.492	0.217	0.689
CR3	0.415	0.557	0.331	0.425	0.424	0.669
CR9	0.377	0.661	0.264	0.463	0.217	0.736
CR10	0.311	0.552	0.216	0.408	0.283	0.784
CR8	0.276	0.562	0.182	0.408	0.228	0.818
CR7	0.131	0.550	0.074	0.392	0.098	0.926
CR6	0.052	0.800	0.027	0.396	0.011	0.973
AR1	0	undef	0	0.383	0	1
CR4	0	undef	0	0.383	0	1
CR5	0	undef	0	0.383	0	1

Table 6.17: Random Forest performance when using entropy metrics on router interface telemetry as the feature set for a 5 minute window

Using a 10-minute window size, the performance in Table 6.18 reveals that AR1 demonstrated the best performance, which was not observed in the NetFlow data set

or the 5-minute window of the telemetry data set. AR1 also achieved the highest F1 score across all window sizes and data sets. However, this high score resulted from the router generating alerts for almost all events, suggesting that it may not understand the attack pattern and instead alerts on all or none of the events. CR2 exhibited a similar performance but with slightly fewer false positives and better precision. The relatively decent performance of CR2 may aid in detecting attacks originating from the attack source side of the network.

Router	F1	Precision	TPR	Accuracy	FPR	FNR
AR1	0.765	0.619	1	0.621	0.989	0
CR2	0.760	0.623	0.973	0.621	0.946	0.027
AR3	0.458	0.564	0.385	0.438	0.478	0.615
CR8	0.451	0.654	0.345	0.483	0.293	0.655
CR9	0.243	0.561	0.155	0.404	0.196	0.845
AR4	0.207	0.692	0.122	0.425	0.087	0.878
VR	0.200	0.562	0.122	0.400	0.152	0.878
CR10	0.198	0.529	0.122	0.392	0.174	0.878
CR11	0.176	0.471	0.108	0.375	0.196	0.892
AR2	0.145	0.667	0.081	0.408	0.065	0.919
CR1	0.134	0.688	0.074	0.408	0.054	0.926
CR7	0.069	0.231	0.041	0.325	0.217	0.959
CR4	0.026	0.667	0.014	0.388	0.011	0.986
CR3	0.013	1	$6.7 \cdot 10^{-3}$	0.388	0	0.993
CR5	0.013	1	$6.7 \cdot 10^{-3}$	0.388	0	0.993
CR6	0	undef	0	0.383	0	1

Table 6.18: Random Forest performance when using entropy metrics on router interface telemetry as the feature set for a 10 minute window

For the 15-minute window size, the performance of AR1 Table 6.19 remained similar to what was observed in the NetFlow data set. The difference is that AR1 now generates alerts for all events, resulting in a significantly higher FPR. AR3 exhibited the second-best performance in this window size, suggesting that it may be more suitable for this type of router. Surprisingly, CR1 lost its ability to detect attacks within this time window, despite performing as the second-best router during the 5-minute window.

Considering the number of green cells for each window size, the best performance appears to be achieved using the 10-minute window size, particularly when maximizing

Router	F1	Precision	TPR	Accuracy	FPR	FNR
AR1	0.756	0.608	1	0.608	1	0
AR3	0.726	0.659	0.808	0.629	0.649	0.192
CR2	0.679	0.604	0.774	0.554	0.787	0.226
CR8	0.510	0.584	0.452	0.471	0.500	0.548
CR10	0.466	0.699	0.349	0.512	0.234	0.651
CR7	0.267	0.706	0.164	0.450	0.106	0.836
AR4	0.261	0.632	0.164	0.433	0.149	0.836
VR	0.225	0.625	0.137	0.425	0.128	0.863
AR2	0.223	0.606	0.137	0.421	0.138	0.863
CR11	0.211	0.559	0.130	0.408	0.160	0.870
CR9	0.130	0.478	0.075	0.388	0.128	0.925
CR3	0.052	0.500	0.027	0.392	0.043	0.973
CR1	0	0	0	0.379	0.032	1
CR4	0	undef	0	0.392	0	1
CR5	0	undef	0	0.392	0	1
CR6	0	undef	0	0.392	0	1

Table 6.19: Random Forest performance when using entropy metrics on router interface telemetry as the feature set for a 15 minute window

the detection of attacks. Furthermore, this window size resulted in the fewest routers being unable to detect any attacks. However, if the emphasis is on the accuracy of alerts, a 5-minute window may be more suitable. It is important to note that none of the window sizes yielded particularly impressive results, indicating that the entropy metrics may not be reliable indicators of attacks.

Combined Classifiers were trained using a combination of the two previously described feature sets. The performance of these classifiers in the 5-minute window on NetFlow records is shown in Table 6.20. It is observed that CR3 performs the best with a higher F1 score compared to the corresponding entropy classifier but lower than the header fields classifier. CR3 shows more false negatives and a lower TPR, indicating that it generally alerts on fewer events. However, it has the lowest FPR among all window sizes while still producing several alerts, implying that most are true positives.

In the case of the 10-minute window, CR3 still performs the best, as seen in Table 6.21. It performs better than with the 5-minute window, and a similar improvement is also observed for VR. When only the entropy feature set is used,

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.748	1.000	0.597	0.963	$8.8 \cdot 10^{-7}$	0.403
VR	0.252	1.000	0.144	0.154	$4.6 \cdot 10^{-4}$	0.856
AR3	0.069	1	0.036	0.616	0	0.964
AR2	0.041	1	0.021	0.858	0	0.979
AR4	$9.4 \cdot 10^{-5}$	1	$4.7 \cdot 10^{-5}$	$5.3 \cdot 10^{-3}$	0	1.000
AR1	$1.6 \cdot 10^{-5}$	1	$8.0 \cdot 10^{-6}$	0.841	0	1.000

Table 6.20: Random Forest performance when using the combination as the feature set for a 5 minute window

VR's performance decreases, suggesting that the combination feature set is more effective for this router. The decrease in performance in the entropy feature set is mainly due to increased false positives, while the combination slightly reduces the false positives. However, for AR2 and AR3, the performance decreases as they fail to detect almost any attacks. This indicates that there is no universally optimal window size for all routers, and parameters should perhaps be chosen individually to achieve better performance.

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.839	1.000	0.723	0.974	$4.7 \cdot 10^{-6}$	0.277
VR	0.329	1.000	0.197	0.207	$4.1 \cdot 10^{-4}$	0.803
AR4	$5.9 \cdot 10^{-4}$	0.998	$2.9 \cdot 10^{-4}$	$5.5 \cdot 10^{-3}$	$9.4 \cdot 10^{-5}$	1.000
AR2	$1.9 \cdot 10^{-5}$	1	$9.6 \cdot 10^{-6}$	0.855	0	1.000
AR3	$1.8 \cdot 10^{-5}$	1	$9.2 \cdot 10^{-6}$	0.602	0	1.000
AR1	$2.4 \cdot 10^{-6}$	1	$1.2 \cdot 10^{-6}$	0.841	0	1.000

Table 6.21: Random Forest performance when using the combination as the feature set for a 10 minute window

The 15-minute window yields the best performance for CR3, as shown in Table 6.22. The F1 score of 96.4% is the highest among all window sizes for the combined feature set. However, the performance of other routers in detecting attacks is poor, with very few detections. Therefore, for routers other than CR3, the 5-minute window is better for detecting the most attacks.

The classifier was also trained on the combined feature set without the inclusion of IP addresses. The performance of this feature set using a 5-minute window is presented in Table 6.23. The F1 score is significantly lower compared to the feature

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.964	1.000	0.930	0.994	$8.8 \cdot 10^{-7}$	0.070
VR	0.063	1.000	0.033	0.044	$3.7 \cdot 10^{-4}$	0.967
AR4	$5.5 \cdot 10^{-4}$	0.999	$2.7 \cdot 10^{-4}$	$5.5 \cdot 10^{-3}$	$4.7 \cdot 10^{-5}$	1.000
AR3	$2.1 \cdot 10^{-5}$	1	$1.0 \cdot 10^{-5}$	0.602	0	1.000
AR2	$1.9 \cdot 10^{-5}$	1	$9.6 \cdot 10^{-6}$	0.855	0	1.000
AR1	$3.0 \cdot 10^{-6}$	1	$1.5 \cdot 10^{-6}$	0.841	0	1.000

Table 6.22: Random Forest performance when using the combination as the feature set for a 15 minute window

set with IP addresses, indicating that the presence of IP addresses influenced the classifier's alerts. In the case of VR, the F1 score is not considerably worse, as it did not perform well with the IP addresses either. This suggests that the classifiers for routers other than CR3 do not effectively learn to recognize malicious IP addresses or attack patterns. This could maybe be attributed to introducing unknown attacks that the classifier probably does not recognize.

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.336	1.000	0.202	0.926	$8.5 \cdot 10^{-6}$	0.798
VR	0.246	1.000	0.140	0.151	$2.3 \cdot 10^{-4}$	0.860
AR3	0.029	0.999	0.015	0.608	$8.1 \cdot 10^{-6}$	0.985
AR2	$2.1 \cdot 10^{-4}$	0.907	$1.0 \cdot 10^{-4}$	0.855	$1.8 \cdot 10^{-6}$	1.000
AR4	$1.1 \cdot 10^{-5}$	1	$5.5 \cdot 10^{-6}$	$5.2 \cdot 10^{-3}$	0	1.000
AR1	0	undef	0	0.841	0	1

Table 6.23: Random Forest performance when using the combination without IP addresses as the feature set for a 5 minute interval

The 10-minute window gives the best F1 score for CR3, as indicated by the green cell in Table 6.24. However, the performance is not particularly good and is significantly worse than when IP addresses are included. AR2 shows similar performance scores for both feature sets, suggesting that IP addresses do not significantly impact the classifier's performance. The lack of performance for AR1, AR2, AR3, and AR4 might be attributed to mislabeled training data or unknown attacks in the testing data set.

The performance of the routers in the 15-minute window is even worse, as shown in Table 6.25. This differs from the feature set with IP addresses, indicating that the

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.458	0.999	0.297	0.935	$2.6 \cdot 10^{-5}$	0.703
VR	0.132	1.000	0.071	0.082	$1.3 \cdot 10^{-4}$	0.929
AR4	$1.8 \cdot 10^{-4}$	0.968	$9.0 \cdot 10^{-5}$	$5.3 \cdot 10^{-3}$	$5.6 \cdot 10^{-4}$	1.000
AR2	$1.9 \cdot 10^{-5}$	1	$9.6 \cdot 10^{-6}$	0.855	0	1.000
AR3	0	0	0	0.602	$8.9 \cdot 10^{-6}$	1
AR1	0	undef	0	0.841	0	1

Table 6.24: Random Forest performance when using the combination without IP addresses as the feature set for a 10 minute interval

few alerts produced in that feature set are attributed to malicious IP addresses.

Router	F1	Precision	TPR	Accuracy	FPR	FNR
CR3	0.457	0.999	0.297	0.935	$1.5 \cdot 10^{-5}$	0.703
VR	0.061	1.000	0.031	0.043	$2.3 \cdot 10^{-4}$	0.969
AR4	$3.3 \cdot 10^{-4}$	0.972	$1.6 \cdot 10^{-4}$	$5.4 \cdot 10^{-3}$	$9.0 \cdot 10^{-4}$	1.000
AR3	0	0	0	0.602	$1.5 \cdot 10^{-5}$	1
AR1	0	undef	0	0.841	0	1
AR2	0	undef	0	0.855	0	1

Table 6.25: Random Forest performance when using the combination without IP addresses as the feature set for a 15 minute interval

Additionally, the classifier was trained on the combination of router interface telemetry measurements and entropy metrics. In the case of the 5-minute window, the victim router achieves the highest F1 score of 0.778, as seen in Table 6.26. It also achieves the highest accuracy of 0.713 across all window sizes. The performance is better than when only the entropy feature set is used but not as good as when only the fields feature set is used. However, the combined feature set exhibits a slightly higher TPR than the fields feature set, suggesting that it is not as effective at distinguishing attack patterns and produces more alerts.

In the 10-minute window, the same thing happens as in the entropy feature set, where AR1 transitions from being the worst-performing router to the best-performing router. However, the performance is better across all metrics compared to the entropy feature set. It also performs better on all metrics compared to the fields feature set, except for the FPR. CR1 is an exception, as it has a low FPR of 5.3%, resulting in a precision of 95%. This router provides more precise alerts but also fewer alerts than

Router	F1	Precision	TPR	Accuracy	FPR	FNR
VR	0.778	0.767	0.790	0.713	0.420	0.210
CR2	0.724	0.648	0.820	0.602	0.782	0.180
CR1	0.669	0.890	0.536	0.662	0.116	0.464
AR3	0.658	0.650	0.665	0.558	0.630	0.335
AR4	0.437	0.569	0.354	0.417	0.471	0.646
AR2	0.423	0.662	0.311	0.460	0.280	0.689
CR11	0.399	0.652	0.288	0.448	0.270	0.712
CR8	0.371	0.607	0.267	0.423	0.304	0.733
CR9	0.356	0.616	0.250	0.423	0.273	0.750
CR3	0.346	0.569	0.248	0.401	0.330	0.752
CR10	0.336	0.653	0.227	0.430	0.212	0.773
CR7	0.138	0.589	0.078	0.378	0.096	0.922
CR5	0.100	1	0.053	0.397	0	0.947
CR6	0.021	1	0.011	0.369	0	0.989
AR1	0	undef	0	0.363	0	1
CR4	0	undef	0	0.363	0	1

Table 6.26: Random Forest performance when using the combination of router interface telemetry measurements as the feature set for a 5 minute interval

expected. The precision and FPR were not as good in the entropy feature set or the fields feature set, indicating that the combination feature set improves alert accuracy for this router.

In the 15-minute window, the precision for CR1 improves even further, as shown in Table 6.28. However, the TPR decreases significantly, potentially making the increase in precision less valuable. Moreover, more routers fail to detect attacks compared to the 10-minute window, which is consistent with the trend observed in the entropy feature set.

Overall, the performance improves when using the combined feature set compared to using only the entropy feature set. Certain metrics on specific routers also show improvement compared to using only the fields feature set.

6.5.15 K-means

For evaluating the performance of K-means clustering, the accuracy of the clustering can be assessed using the TPR, as depicted in the heatmap shown in Figure 6.61. The results indicate that using only the header fields and the combined feature set

Router	F1	Precision	TPR	Accuracy	FPR	FNR
AR1	0.802	0.669	1	0.675	0.952	0
CR2	0.795	0.662	0.994	0.662	0.976	$6.3 \cdot 10^{-3}$
VR	0.741	0.700	0.787	0.638	0.648	0.213
AR3	0.691	0.697	0.685	0.597	0.574	0.315
CR1	0.671	0.950	0.519	0.665	0.053	0.481
CR8	0.460	0.671	0.350	0.459	0.331	0.650
AR2	0.305	0.573	0.207	0.377	0.297	0.793
CR9	0.257	0.533	0.169	0.356	0.284	0.831
CR10	0.184	0.581	0.109	0.362	0.152	0.891
CR11	0.154	0.561	0.089	0.354	0.135	0.911
AR4	0.136	0.756	0.075	0.375	0.046	0.925
CR7	0.122	0.406	0.072	0.320	0.201	0.928
CR3	0.035	1	0.018	0.354	0	0.982
CR4	0.017	0.627	$8.8 \cdot 10^{-3}$	0.344	0.010	0.991
CR5	0.013	1	$6.3 \cdot 10^{-3}$	0.346	0	0.994
CR6	0	undef	0	0.342	0	1

Table 6.27: Random Forest performance when using the combination of router interface telemetry measurements as the feature set for a 10 minute interval

in a 15-minute window on the NetFlow data set exhibit nearly perfect clustering of attacks into the same cluster. The other combined feature sets on the NetFlow data set also demonstrate high TPR across all routers except the victim router. In contrast, the clustering performance on the telemetry data set displays greater variation and appears to be more dependent on the specific router rather than the method. Like the RF classifier, the entropy metrics exhibit inferior performance on both data sets, suggesting that these features lack sufficient information for effective clustering.

However, achieving high TPR comes at the expense of a high FPR. For routers with near-perfect TPR, the FPR is consistently above 80%. This observation could imply that most measurements are clustered into a single cluster. Notably, the victim router utilizing the header field feature set demonstrates a relatively lower FPR of 55.7%. This discrepancy may be attributed to the fact that the victim router has much less benign traffic going through it than, e.g., AR2, so the attack traffic was the majority of the traffic, so it probably was clustered somewhat together.

Regarding precision, the performance of alerts on the NetFlow data set, except

Router	F1	Precision	TPR	Accuracy	FPR	FNR
AR1	0.803	0.671	1	0.671	1	0
AR3	0.783	0.718	0.860	0.679	0.689	0.140
VR	0.757	0.703	0.821	0.647	0.706	0.179
CR2	0.740	0.692	0.795	0.625	0.722	0.205
CR8	0.525	0.633	0.449	0.456	0.531	0.551
CR1	0.509	0.958	0.347	0.552	0.031	0.653
CR10	0.506	0.892	0.353	0.537	0.087	0.647
CR11	0.347	0.707	0.230	0.420	0.194	0.770
AR4	0.219	0.647	0.132	0.370	0.147	0.868
CR9	0.209	0.575	0.127	0.352	0.191	0.873
AR2	0.159	0.706	0.090	0.364	0.076	0.910
CR3	0.097	0.677	0.052	0.347	0.051	0.948
CR7	0.012	1	$6.2 \cdot 10^{-3}$	0.333	0	0.994
CR4	0	undef	0	0.329	0	1
CR5	0	undef	0	0.330	0	1
CR6	0	undef	0	0.329	0	1

Table 6.28: Random Forest performance when using the combination of router interface telemetry measurements as the feature set for a 15 minute interval

for AR4 and VR, is generally subpar. These two routers achieve precision levels of approximately 90% using the combined feature sets and the header fields feature set. However, as evident in Figure 6.61, VR only achieves a TPR of around 70% for two of the combined feature sets, indicating that high precision does not necessarily correspond to detecting all attacks. The entropy feature set with a 15-minute window yields the highest precision for the remaining routers, although it was only 50% on average.

Regarding the telemetry data set, precision is generally higher, but this improvement comes at the expense of TPR. The telemetry data set tends to miss more attacks; however, the alerts are more accurate when attacks are detected. In this case, the fields feature set and the combined feature set with 5 and 10-minute windows exhibit the best performance, aligning with their more promising TPR. AR3 and CR3 consistently demonstrate inferior performance across most metrics, except for FPR. As mentioned in Section 6.3.15, this may be attributed to the clustering approach primarily segregating high-volume traffic from other traffic. The telemetry measurements in this router predominantly cluster into the non-attack category due to consistently low Davies–Bouldin scores and the largest cluster being the least

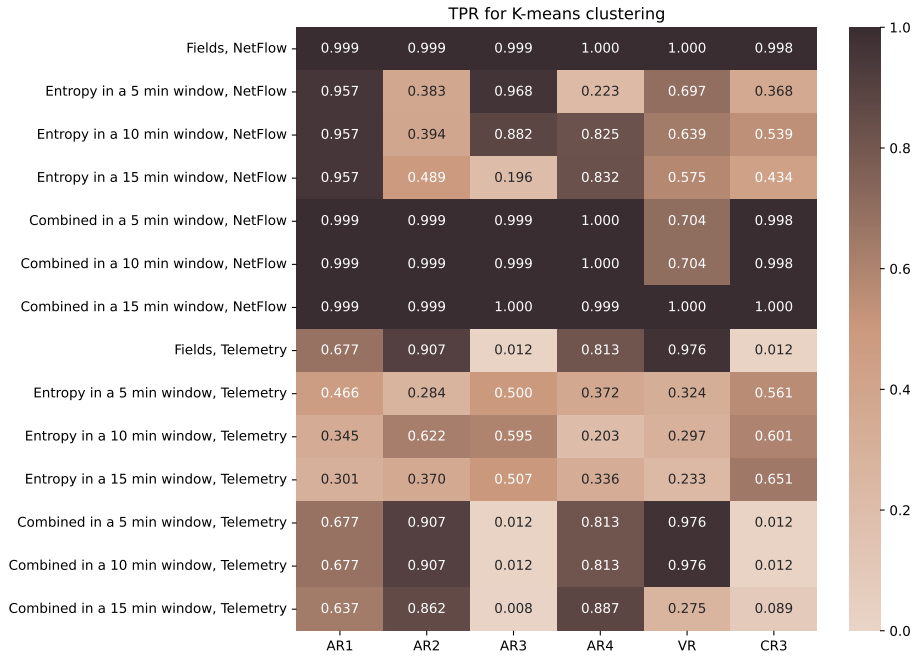


Figure 6.61: The TPR for all routers of K-means clustering on both data sets during the third round of attack generation

compact, indicating it as benign.

Consequently, the attack cluster becomes relatively small, and whether the measurements within the cluster correspond to attack periods becomes arbitrary. On average, CR6 exhibits the best performance across most metrics. This router serves as one of the core routers through which attacks from attacker 4 go through on their way to the victim. Unlike CR3 and AR3, this router does not merely segregate measurements based on volume but demonstrates more effective clustering. The superior performance of CR6 may be attributed to the more significant variability in traffic patterns, preventing measurements from becoming too similar and being clustered into a single cluster.

6.6 Alert fusion

After examining individual detection methods' performance, the focus now shifts to evaluating their performance when combined. This section will try to answer the second research question, denoted as **R2** in Section 1.3. As mentioned in Section 5.1,

the fusion process comprises five distinct components. Alerts are fused based on various factors, including *time*, *attack type*, *packet size distribution*, and *IP addresses*. Subsequently, the combined outputs from these fusion units are ranked. Each unit merges a set of alerts and generates a new alert summarizing the aggregated information, which is then forwarded to the ranking unit. Precision is primarily considered to assess the fusion units' performance, as one of the fusion's objectives could be to present more accurate alerts. Furthermore, the unavailability of the actual number of negatives in this part of the IDS necessitates relying solely on precision as the performance metric. This unavailability comes from the fact that the number of negatives would also include all the negatives from each detection method.

This section presents the results for the six tests conducted, defined in Section 6.5. Each test used one of the three different sliding window sizes: 5, 10, or 15 minutes. Additionally, each test used one of two methods to determine the thresholds: either by maximizing the TPR or the F1 score from the previous round of ground truth generation. The results are presented in tables, with the leftmost column indicating the window size and threshold generation method used.

6.6.1 Time

The fusion based on time is the first unit under examination. Table 6.29 displays the precision of the alerts before and after passing through this unit. Additionally, the precision of the outgoing combined alerts is calculated, determining their status as either false or true positives based on the majority ground truth within the summary. For instance, if a combined alert is formed by aggregating 1000 false positives and 600 true positives, it would be considered a false positive in the combined alert. The table shows a slight improvement in precision for all fusion tests, except when utilizing F1 scores to establish a threshold with a 5-minute window. In this case, fusion results in decreased precision. This decline might be attributed to a threshold implemented within the unit, which determines whether a summary alert should be generated. This threshold was introduced to minimize the number of false alerts. As depicted in Table 6.30, the number of alerts is reduced by at least half. However, this unit reduces true positives to a greater extent than false positives, which is not ideal. The exception is when employing 15-minute windows.

Notably, utilizing a 15-minute window size yields improved precision and reduction outcomes. One possible explanation for this observation could be that fewer alerts are generated by the detection methods when employing this window size. This occurrence might be linked to the fact that K-means clustering operates on a 30-minute clustering interval instead of 15 minutes, which could impact the size of the attack cluster. However, as discussed in Section 6.4, numerous alerts are generated by the detection methods, implying that the MQTT implementation cannot process

	Incoming	Outgoing	Outgoing combined
TPR			
5 min	0.317	0.295	0.327
10 min	0.316	0.293	0.333
15 min	0.323	0.356	0.333
F1			
5 min	0.319	0.310	0.314
10 min	0.311	0.297	0.331
15 min	0.314	0.432	0.371

Table 6.29: Precision scores for fusion on time

all of them, as the aggregation unit accounts for slightly over 10% of the total alerts. It is also plausible that some detection methods crashed during testing, making them unable to deliver all their alerts. The MQTT broker discards messages from its queue upon reaching a limit. The default limit for a Mosquitto broker is 1000, so if the detection methods generated over 1000 messages within a short period, the broker might have dropped them, resulting in the alerts not being processed by the alert fusion component of the IDS. Overcoming these issues could have involved rewriting the algorithm; however, time constraints prevented this, as these limitations were identified late in the thesis work. Alternatively, a decentralized fusion approach could have been adopted, employing one MQTT broker for each router, which would send summarized alerts to a central broker for further fusion. This approach would enhance the scalability of the IDS. Another consideration could have been excluding the K-means clustering detection methods, as they account for over 90% of all the alerts while not rendering outstanding results.

6.6.2 Attack types

Proceeding to the fusion of alerts based on attack types, considerably better precision scores are observed than the fusion based on time, as illustrated in Table 6.31. This improvement may stem from the improved incoming precision scores. Attack types for the K-means clustering alerts were omitted due to the absence of a clear means to determine the specific attack type based on the Davies-Bouldin score, as discussed in Section 6.3.15 and Section 6.5.15. Consequently, the number of alerts in this unit was reduced by 99% compared to the time fusion.

In Table 6.31, the outgoing precision consistently performs well and surpasses the

TPR	True Positives	False Positives
5 min	0.537	0.487
10 min	0.520	0.466
15 min	0.388	0.471
F1		
5 min	0.520	0.499
10 min	0.533	0.501
15 min	0.344	0.606

Table 6.30: Alert reduction for fusion on time

incoming precision. However, the combined outgoing precision is almost always worse than even the incoming precision. This discrepancy could arise from the limited number of combined alerts generated, causing each alert to have a more significant impact on the performance. The significant difference between the total outgoing and combined precision suggests that when the number of false positives exceeded the threshold, they were likely summarized together, resulting in a combined false positive. Additionally, during periods with numerous true positives, they might have been merged into a single alert, diminishing their contribution to the precision metric. Addressing this issue could involve assigning weights to the combined precision based on how big the difference was between true and false positives.

TPR	Incoming	Outgoing	Outgoing combined
5 min	0.785	0.991	0.875
10 min	0.775	0.951	0.697
15 min	0.775	0.806	0.596
F1			
5 min	0.797	1.000	1.000
10 min	0.795	0.846	0.637
15 min	0.821	0.866	0.627

Table 6.31: Precision scores for fusion on attack type

Table 6.32 illustrates the reduction in alerts for this unit. All fusion tests reduce the number of false positives by over 80%. However, the true positives also experience substantial reduction. The most favorable scenario occurs with the 5-minute window using TPR as the threshold basis, exhibiting the largest difference between the reduction in positives. Nonetheless, losing 88% of true positives is far from desirable. This outcome suggests that either the threshold for combining alerts was too high or the differentiation between true and false positives proved challenging. As observed in the ROC curves for the detection methods, most methods achieved an AUC of approximately 0.5, indicating the arbitrary nature of establishing an accurate threshold.

	True Positives	**False Positives**
TPR		
5 min	0.876	0.996
10 min	0.875	0.978
15 min	0.809	0.842
F1		
5 min	0.916	1.000
10 min	0.881	0.916
15 min	0.927	0.948

Table 6.32: Alert reduction for fusion on attack type

6.6.3 Packet size distribution

The alert fusion unit focusing on packet size distribution outperforms the time fusion, as depicted in Table 6.33. The incoming precision is comparatively inferior to the unit based on attack types. However, the outgoing precision is remarkably the highest observed thus far on average. Similarly, the precision is optimal when employing a 5-minute window and using F1 scores as the threshold basis. Notably, in this case, the improved performance does not appear to come only from the incoming precision, as the incoming precision is superior when using TPR. This observation might suggest that the fusion unit is effectively functioning as intended.

From examining the reduction in alerts for the fusion based on packet size distributions (Table 6.34), it is evident that most tests yield significant reductions in false positives while retaining the majority of true positives. The only exception is the 10-minute window using F1 scores, which exhibits a higher reduction in true

TPR	Incoming	Outgoing	Outgoing combined
5 min	0.698	0.953	0.925
10 min	0.440	0.830	0.821
15 min	0.562	0.896	0.868
F1			
5 min	0.651	0.955	0.925
10 min	0.533	0.936	0.952
15 min	0.697	0.906	0.903

Table 6.33: Precision scores for fusion on packet size distribution

positives but still keeps most of the alerts. The most favorable ratio between true and false positive reduction is observed when employing a 5-minute window and utilizing F1 scores, coinciding with the highest precision. As the number of different packet sizes in a 5-minute window is probably less than in bigger window sizes, it would make sense that comparing this to other routers would be easier.

TPR	True Positives	False Positives
5 min	0.184	0.906
10 min	0.209	0.872
15 min	0.235	0.887
F1		
5 min	0.166	0.927
10 min	0.482	0.960
15 min	0.263	0.824

Table 6.34: Alert reduction for fusion on packet size distribution

6.6.4 Ranking

The combined alerts generated by the fusion units are subsequently processed by the ranking unit as depicted in Figure 5.1 where this unit is the last step in the

IDS. Subsequently, the ranking unit ranks them based on either attack type and deviation score or solely on the deviation score. Both approaches were evaluated to determine the most effective results. Table 6.35 presents the precision of the incoming combined alerts obtained from the fusion units. The overall performance is subpar, potentially influenced by the low precision of the time fusion unit. The highest score is observed when utilizing a 15-minute window with the F1 score as the threshold basis. Notably, this test also yielded the best performance in the time fusion unit, indicating a strong correlation between incoming precision and the performance of this unit. Furthermore, this test exhibited the lowest reduction ratio in the fusion of attack types, which is not encouraging. It is also worth noting that no other units than the time unit performed best during this test.

	Incoming combined
TPR	
5 min	0.521
10 min	0.399
15 min	0.502
F1	
5 min	0.499
10 min	0.488
15 min	0.559

Table 6.35: Precision scores incoming alerts into the ranking unit

The ranking unit ranks the combined alerts it received in the last 15 minutes based on what it perceives as the most dangerous threats. An example of such a ranking is illustrated in Figure 6.62. Each ranking includes the ranked position of each combined alert, start and end times, the routers involved, mean and standard deviation of the deviation scores, potential attack types, and ground truth labels for performance evaluation. This ranking is made during the SlowLoris attack in round three of the ground truth generation. The ranking is based on the deviation scores, with the highest score receiving the top position.

The performance of the ranking unit for the six tests was calculated using the ground truth labels. Table 6.36 presents the precision of high-ranking alerts, defined as the top 10 alerts. Three precision scores were calculated by emphasizing different factors. The “Majority wins” approach means that for each combined alert in the ranking, the majority of the ground truth labels decide if it is a true or false positive.

Position	Start time	End time	Routers involved	Deviation score	Attack type	Real labels
0	2023-03-24T14:26:00Z	2023-03-24T14:27:00Z	VR, AR1, CR3, AR2, AR3	Mean:33.30014, Standard deviation:68.68651	No type: 0.01468, SYN Flood: 0.01313, Same protocol: 0.00552, Flooding: 0.00394	0: 62247, 1: 71
1	2023-03-24T14:27:00Z	2023-03-24T14:28:00Z	VR, AR1, CR3, AR2, AR3	Mean:26.47317, Standard deviation:76.82008	Flooding: 0.00903, No type: 0.00888, SYN Flood: 0.00841, Same protocol: 0.00552, Different protocols: 0.00126	0: 41943, 1: 93
2	2023-03-24T14:19:00Z	2023-03-24T14:20:00Z	VR, AR1, CR3, AR2, AR3	Mean:21.53323, Standard deviation:42.62564	Same protocol: 0.01309, Different protocols: 0.00652, Flooding: 0.00585, SYN Flood: 0.00440, No type: 0.00317	0: 32372, 1: 57
3	2023-03-24T14:25:00Z	2023-03-24T14:26:00Z	VR, AR1, CR3, AR2, AR3	Mean:19.72079, Standard deviation:39.88408	Different protocols: 0.01370, No type: 0.01367, SYN Flood: 0.00922, Flooding: 0.00396, Same protocol: 0.00126	0: 37992, 1: 55
4	2023-03-24T14:20:00Z	2023-03-24T14:21:00Z	VR, AR1, CR3, AR2, AR3	Mean:19.28287, Standard deviation:45.89803	Same protocol: 0.01389, Flooding: 0.00963, No type: 0.00817, Different protocols: 0.00652, SYN Flood: 0.00388	0: 46003, 1: 76
5	2023-03-24T14:21:00Z	2023-03-24T14:22:00Z	VR, AR1, CR3, AR2, AR3	Mean:16.46956, Standard deviation:27.15231	Flooding: 0.01619, No type: 0.01091, SYN Flood: 0.00550, Different protocols: 0.00469	0: 31178, 1: 52
6	2023-03-24T14:18:00Z	2023-03-24T14:19:00Z	VR, AR1, CR3, AR2, AR3	Mean:13.23159, Standard deviation:25.74369	Flooding: 0.01585, Different protocols: 0.00652, SYN Flood: 0.00550, No type: 0.00319	0: 39116, 1: 79
7	2023-03-24T14:22:00Z	2023-03-24T14:23:00Z	VR, AR1, CR3, AR2, AR3	Mean:12.36439, Standard deviation:16.66650	No type: 0.01290, Flooding: 0.01273, SYN Flood: 0.00566, Different protocols: 0.00469, Same protocol: 0.00126	0: 35064, 1: 55
8	2023-03-24T14:24:00Z	2023-03-24T14:25:00Z	VR, AR1, CR3, AR2, AR3	Mean:12.21308, Standard deviation:20.14405	No type: 0.00929, SYN Flood: 0.00679, Different protocols: 0.00675, Flooding: 0.00647, Low-Rate: 0.00545, Same protocol: 0.00126	0: 41388, 1: 54
9	2023-03-24T14:17:00Z	2023-03-24T14:18:00Z	VR, AR1, CR3, AR2, AR3	Mean:11.93405, Standard deviation:20.52121	Flooding: 0.01199, Different protocols: 0.00730, No type: 0.00698, SYN Flood: 0.00498, Same protocol: 0.00409	0: 32261, 1: 57
10	2023-03-24T14:15:00Z	2023-03-24T14:16:00Z	VR, AR1, CR3, AR2, AR3	Mean:10.08632, Standard deviation:17.32756	Flooding: 0.01536, Different protocols: 0.01282, No type: 0.00966, SYN Flood: 0.00469	0: 26858, 1: 47
11	2023-03-24T14:16:00Z	2023-03-24T14:17:00Z	VR, AR1, CR3, AR2, AR3	Mean:10.05241, Standard deviation:19.38975	Different protocols: 0.01282, No type: 0.00670, Flooding: 0.00667, SYN Flood: 0.00339, Low-Rate: 0.00066	0: 26925, 1: 89
12	2023-03-24T14:23:00Z	2023-03-24T14:24:00Z	VR, AR1, CR3, AR2, AR3	Mean:8.36625, Standard deviation:15.97071	Flooding: 0.01408, No type: 0.01047, Different protocols: 0.00540, Low-Rate: 0.00391, SYN Flood: 0.00388, Same protocol: 0.00258	0: 47104, 1: 85
13	2023-03-24T14:22:00Z	2023-03-24T14:23:00Z	AR3, CR3	Mean:1.72942, Standard deviation:2.21915	Different protocols: 0.00550, Same protocol: 0.00126	1: 6
14	2023-03-24T14:22:00Z	2023-03-24T14:23:00Z	AR3, AR2	Mean:1.54249, Standard deviation:1.92460	Different protocols: 0.00428, Same protocol: 0.00323	1: 8
15	2023-03-24T14:23:00Z	2023-03-24T14:24:00Z	AR3, CR3	Mean:1.35856, Standard deviation:1.45784	Different protocols: 0.01364, Same protocol: 0.00505	1: 18
16	2023-03-24T14:22:00Z	2023-03-24T14:23:00Z	AR3, AR2	Mean:1.29796, Standard deviation:1.79441	Different protocols: 0.00560, Same protocol: 0.00389	1: 10
17	2023-03-24T14:23:00Z	2023-03-24T14:24:00Z	AR3, AR2, CR3	Mean:1.16384, Standard deviation:1.68679	Different protocols: 0.01478, Same protocol: 0.00514	1: 19
18	2023-03-24T14:23:00Z	2023-03-24T14:24:00Z	AR3, AR2, CR3	Mean:1.15675, Standard deviation:1.63541	Different protocols: 0.01611, Same protocol: 0.00576	1: 21
19	2023-03-24T14:23:00Z	2023-03-24T14:24:00Z	AR3, AR2, CR3	Mean:1.04448, Standard deviation:1.56091	Different protocols: 0.01886, Same protocol: 0.00641	1: 24
20	2023-03-24T14:22:00Z	2023-03-24T14:23:00Z	AR3, AR2, CR3	Mean:1.03950, Standard deviation:1.64316	Different protocols: 0.01621, Same protocol: 0.00515	1: 20
21	2023-03-24T14:23:00Z	2023-03-24T14:24:00Z	AR3, AR2, CR3	Mean:0.96170, Standard deviation:1.49215	Different protocols: 0.02151, Same protocol: 0.00707	1: 27
22	2023-03-24T14:22:00Z	2023-03-24T14:23:00Z	AR3, AR2, CR3	Mean:0.95104, Standard deviation:1.60429	Different protocols: 0.01488, Same protocol: 0.00453	1: 18
23	2023-03-24T14:24:00Z	2023-03-24T14:25:00Z	AR3, CR3	Mean:0.84727, Standard deviation:0.81096	Different protocols: 0.02726, Same protocol: 0.01009	1: 28, 0: 0
24	2023-03-24T14:24:00Z	2023-03-24T14:25:00Z	AR3, CR3	Mean:0.84042, Standard deviation:0.81624	Different protocols: 0.02045, Same protocol: 0.00757	1: 21, 0: 6
25	2023-03-24T14:24:00Z	2023-03-24T14:25:00Z	AR3, CR3	Mean:0.78678, Standard deviation:1.20751	Different protocols: 0.01743, Same protocol: 0.00316	1: 17, 0: 1
26	2023-03-24T14:24:00Z	2023-03-24T14:25:00Z	AR3, CR3	Mean:0.77442, Standard deviation:1.16073	Different protocols: 0.01875, Same protocol: 0.00378	1: 18, 0: 2
27	2023-03-24T14:24:00Z	2023-03-24T14:25:00Z	AR3, CR3	Mean:0.71677, Standard deviation:1.12176	Different protocols: 0.02151, Same protocol: 0.00378	1: 20, 0: 2
28	2023-03-24T14:24:00Z	2023-03-24T14:25:00Z	AR3, CR3	Mean:0.66860, Standard deviation:1.08594	Different protocols: 0.02415, Same protocol: 0.00378	1: 22, 0: 2
29	2023-03-24T14:27:00Z	2023-03-24T14:28:00Z	AR3, CR3	Mean:0.52304, Standard deviation:0.24864	Different protocols: 0.01452, Same protocol: 0.00950	1: 18, 0: 2
30	2023-03-24T14:22:00Z	2023-03-24T14:23:00Z	VR, AR3, AR2	Mean:0.51685, Standard deviation:0.16542	Different protocols: 0.00277, Same protocol: 0.00066	1: 3
31	2023-03-24T14:27:00Z	2023-03-24T14:28:00Z	AR3, CR3	Mean:0.51443, Standard deviation:0.21459	Different protocols: 0.01107, Same protocol: 0.00675	1: 14, 0: 2
32	2023-03-24T14:27:00Z	2023-03-24T14:28:00Z	AR3, CR3	Mean:0.49166, Standard deviation:0.22546	Different protocols: 0.01320, Same protocol: 0.00818	1: 16, 0: 2
33	2023-03-24T14:27:00Z	2023-03-24T14:28:00Z	AR3, CR3	Mean:0.48061, Standard deviation:0.19216	Different protocols: 0.01055, Same protocol: 0.00613	1: 13, 0: 1
34	2023-03-24T14:25:00Z	2023-03-24T14:26:00Z	AR3, CR3	Mean:0.44824, Standard deviation:0.55257	Different protocols: 0.02929, Same protocol: 0.00395	1: 25, 0: 2
35	2023-03-24T14:25:00Z	2023-03-24T14:26:00Z	AR3, CR3	Mean:0.42671, Standard deviation:0.52084	Different protocols: 0.03194, Same protocol: 0.00527	1: 28, 0: 2
36	2023-03-24T14:27:00Z	2023-03-24T14:28:00Z	AR3, CR3	Mean:0.39964, Standard deviation:0.16748	Same protocol: 0.00275, Different protocols: 0.00259	1: 5

Figure 6.62: An example of how a ranking on deviation score looks like

For instance, in Figure 6.62, the top alert had 62 247 false positives and 71 true positives, counting as a single false positive. The false and true positives for each alert in the top 10 rankings are tallied to calculate the precision. The weighted version of "Majority wins" follows the same principle, but the weight of each alert affects its contribution to the precision instead of all alerts contributing the same. So if an alert has a high weight, it counts more towards the precision than a low-weight alert. This weighting indicates the likelihood of the alert being a true alert. The combined alerts could also have been ranked using these weights, which would have given a better performance in the "weighted majority wins" approach of calculating the precision.

"At least one" refers to the fact that at least one of the ground truth labels in the combined alert was positive. Thus, all combined alerts with at least one true positive are counted as true positives in the final precision. This approach yielded the highest precision, as it only requires one true positive per alert. Nevertheless, the precision remains imperfect, with a maximum value of 80.3% achieved when ranking alerts based on deviation scores within a 5-minute time window using F1 scores as the threshold basis. Even during periods without attacks, the ranking unit might still produce rankings, as it ranks the alerts received in the last 15 minutes at a frequency of one minute if it has received a new alert in the last minute. The

rationale behind this approach is to provide the IDS user with updated rankings every minute for assessing the situation based on the top alerts. So during periods without attacks, there will still be a ranking, but the user can estimate the severity of the situation by observing the deviation score's magnitude.

The "majority wins" approach for calculating precision does not yield satisfactory results. The highest score, 57.1%, is obtained when ranking alerts based on deviation scores using F1 scores as the threshold and a window size of 10. Alternative approaches to ranking alerts were explored, such as using weights or dividing the mean of the deviation score by the standard deviation. When ranking alerts based on the modified deviation score defined as $\frac{\text{mean of } D}{\text{standard deviation of } D}$, where D represents the original deviation score, the "majority wins" approach achieves the highest precision among all ranking methods, reaching a maximum score of 68%. It also outperforms other rankings methods in the "weighted majority wins" approach for three tests, with scores around 60%.

Regarding the "at least one" approach, the ranking based on the original deviation scores delivers the best results. This outcome can be attributed to some alerts with high deviation scores containing numerous false positives, contributing to the deviation score. Consequently, these alerts exhibit a large standard deviation, causing them to rank lower.

Overall, the ranking based on deviation scores proves to be the most effective among the two original rankings, which were either the attack types or the deviation scores. If the emphasis is placed on the majority of alerts forming the combined alerts being true, ranking the combined alerts based on $\frac{\text{mean of } D}{\text{standard deviation of } D}$ is recommended. Similarly, if the emphasis is placed on that the alerts should be probable attacks based on knowledge of detection method precision from previous attacks, ranking the alerts based on weight is a viable option. However, if weights are used for ranking alerts, relying on TPR for result evaluation is advisable. Otherwise, the mean divided by the standard deviation yields better results in the "weighted majority wins" approach when F1 scores are used to set the thresholds.

Combining different anomaly detection methods offers the benefit of significantly reducing the number of alerts in the ranking output while achieving an okay precision for high-ranking alerts. This argument is supported by Table 6.5, which demonstrates that the ranking unit presents a mere $87.6 \cdot 10^{-5}\%$ of the total alerts while successfully alerting on 10 out of 11 attack types (see last row of heatmap in Figure 6.48).

	Majority wins	Majority wins; weighted	At least one
TPR			
Deviation scores			
5 min	0.248	0.295	0.741
10 min	0.298	0.223	0.801
15 min	0.532	0.350	0.796
Attack types			
5 min	0.253	0.431	0.716
10 min	0.262	0.189	0.790
15 min	0.514	0.285	0.783
F1			
Deviation scores			
5 min	0.306	0.407	0.803
10 min	0.571	0.327	0.798
15 min	0.497	0.334	0.725
Attack types			
5 min	0.227	0.366	0.779
10 min	0.491	0.266	0.772
15 min	0.469	0.279	0.699

Table 6.36: Precision scores for ranking

6.7 Summarizing remarks

This section will provide a summary of the research findings, along with the key insights derived from the results.

Concerning research question **R1**, the impact of different anomaly detection methods is influenced by various factors. The outcomes of these methods depend on factors such as the specific router where data was collected, the type of DDoS attack launched, the thresholds employed, how the methods were implemented, and other related factors. The detection methods can be categorized into three

groups based on their effectiveness. Category one encompasses methods where the effectiveness cannot be conclusively determined. Category two comprises methods that demonstrate effectiveness depending on certain factors. The final category includes methods that exhibit clear effectiveness in detecting DDoS attacks.

Effectiveness is not clear Unfortunately, most of the detection methods fall into this category. The lack of clarity in determining their effectiveness is due to the absence of clear indicators based on the available results. Some methods showed promise in specific attack scenarios, such as the threshold on ICMP packets during ping flood and R.U.D.Y attacks. However, their overall effectiveness remains uncertain as they did not perform adequately in the case of the ping flood attack alone. The entropy-based measurements also fall into this category as they did not exhibit a clear ability to detect attacks.

Effectiveness depends This category encompasses detection methods that demonstrate decent performance depending on various factors. The empirical mean-variance model's effectiveness is heavily influenced by the router on which it is implemented. Only VR showed consistently effective performance across all features. However, for features involving packets, the model also exhibited effectiveness in the case of CR1. This might suggest that the model could detect attacks using features involving packets up to two hops away from the victim.

The threshold on ICMP destination unreachable packets also depends on the router, with attack routers showing promising performance. The choice of window size for the sliding window also impacts the effectiveness of this method. Smaller window sizes generally yield better performance. The threshold selection also significantly affects the method's precision, as illustrated in Figure 6.48, where the precision of this method is far poorer than its potential.

The machine learning algorithms also ended up in this category, as their performance varies based on the availability of ideal inputs. The RF classifier performs well only on CR3 in the NetFlow records and AR1 or VR in the router interface telemetry data. However, this performance is contingent upon factors such as the utilization of IP addresses for training the model, the feature set used, the window size, and the inclusion of the attack type in the training set.

The performance of K-means clustering is also heavily influenced by the router and the specific attack type during a given period. It demonstrates better performance in detecting high-volume attacks in less congested routers. The entropy feature set performs poorly, except during the SYN flood attack, indicating that the effectiveness of clustering is closely linked to the type of DDoS attack.

Effectiveness is clear The only detection method that clearly demonstrates effectiveness in detecting attacks is the method that checks for the simultaneous setting of the URG, PSH, and FIN flags. This method achieved 100% precision during the Xmas attack, which was the intended target of the detection.

Regarding research question **R2**, the benefits of alert fusion depend on the preferences and objectives of the IDS user. Alert fusion proves effective if the objective is to significantly reduce the number of alerts while still detecting 10 out of 11 attacks. In such cases, the additional benefit is a more manageable number of alerts ranked based on their perceived level of danger. However, if the goal is to retain all true positive alerts, the overall effectiveness of alert fusion is determined by the precision of the combined detection methods. Another benefit of alert fusion is that the combined alerts in the ranking provide an overview of the involved routers and the potential attack types associated with the triggered event. These combined alerts also include a deviation score indicating the alert's severity. This aids the IDS user in gaining a global view of the situation, enabling them to allocate resources more effectively when mitigating potential attacks.

6.7.1 Challenges

Several challenges were encountered during the process of conducting this master's thesis. Among these challenges, the most significant was due to the privacy concerns surrounding the NetFlow records. Due to the inclusion of sensitive information such as IP addresses in these records, it was necessary to execute the detection methods on a machine owned by Sikt to obtain the outputs. However, this machine had certain hardware limitations, which resulted in a tedious process. Moreover, the privacy issue prevented the execution of the IP address-based alert fusion on the authors' machine in conjunction with the other units. To obtain results from this fusion unit, a complete reconstruction of the algorithm and its input processing methodology would have been required to enable execution on the machine at Sikt. Unfortunately, time constraints prevented the implementation of such changes.

Regrettably, there was insufficient time to analyze the outputs from attack number four. This was very unfortunate as it could have given additional insights into the performance of the detection methods.

Additionally, a considerable challenge arose from the extensive codebase accumulated throughout the master's thesis. The codebase encompassed nearly 40 000 lines of code and consisted of 52 individual detection methods. Consequently, implementing changes that applied to all detection methods became time-consuming, necessitating substantial code refactoring on each occasion.

Chapter 7
Conclusion

In this thesis, the effect of different anomaly detection methods in detecting various DDoS attacks and the potential benefits of their combination was investigated. A total of 52 individual detection methods were examined, each utilizing one of two data sets to detect 11 different DDoS attacks. These methods were integrated into an IDS that employed fusion techniques based on time, attack type, and packet size distribution to combine the outputs of the individual methods. Subsequently, the fused alerts were ranked according to their perceived level of danger. Effectiveness was evaluated by comparing the results with the ground truth obtained from three rounds of generated DDoS attacks in a real ISP network.

The findings suggest that certain detection methods exhibited promising results depending on the specific testing environment. However, based on the results, most methods did not demonstrate clear effectiveness. The best results were observed when the data originated from less congested routers or routers close to the victim, which is to be expected as the attack traffic would be most noticeable in these routers. Furthermore, the effectiveness of the detection methods was significantly influenced by the chosen threshold for detecting attacks, underscoring the importance of careful threshold selection based on regularly updated data.

Among the examined attacks, the SYN flood attack was generally the easiest to detect across the detection methods, while the LRDDoS attacks proved to be the most challenging. Among the LRDDoS attacks, the Slow Read attack was the easiest to detect, although the effectiveness of the methods for detecting this attack was still inconclusive.

The overall performance scores must be interpreted cautiously, as unknown attacks could be present in the traffic data. Additionally, the labeling accuracy of the router interface telemetry data set, based solely on the timing of attacks, introduces further uncertainty. These factors complicate the conclusive assessment of the detection methods' performance. However, if the labeling accuracy of the router interface

telemetry data set is deemed reliable, certain methods exhibited decent performance. A possible solution for the ISP could be to use router interface telemetry data set-based detection methods as triggers for further investigation using the NetFlow records. This approach, suggested by previous researchers [47], considers that the NetFlow data set requires more storage and processing resources. This approach will be a trade-off between having better detection and including all methods or saving resources and only processing the NetFlow records when further inspection is needed.

Another trade-off the ISP could evaluate is the balance between sampling many flows on few interfaces or fewer flows on more interfaces. As mentioned in Section 6.3.14, attacks like the LRDDoS attacks were not visible in more trafficked routers. The empirical mean-variance model showed that detection of attacks two hops away from the victim was possible, so sampling some flows on the core routers may have enabled the detection of the LRDDoS attacks.

Regarding the alert fusion results, it is concluded that the additional benefits depend on what the user of the IDS values. If the objective is to reduce the number of alerts and prioritize ease of management, the fusion technique achieves this goal. However, this reduction in alerts leads to the loss of many true positive detections. Nonetheless, the fusion-based IDS detected 10 out of 11 attacks, highlighting its efficacy. One potential benefit of the alert fusion is the overview of what routers were involved in a potential attack and what attack it might have been. If the user values this summarized view of alerts, the alert fusion does provide this additional benefit.

7.1 Future work

There are many different ways this implementation of an IDS can be improved in future work. One major enhancement involves enabling real-time operation, which necessitates enhancing the effectiveness of all detection methods. With the right processing power and architectural modifications, achieving real-time detection could be feasible, particularly for the detection methods utilizing the router interface telemetry data. Additionally, incorporating a graphical user interface (GUI) that displays the ranking of alerts would enhance the usability of the IDS.

Expanding the IDS by incorporating additional data sets, such as DNS requests to the ISP's DNS servers, can further improve the detection of DDoS attacks like DNS Reflection Amplification. Detecting this specific attack is relevant, considering its prevalence, accounting for nearly 30% of attacks in the first quarter of 2023 [1].

Future work could also explore replacing ineffective methods with other promising ones. A vital consideration derived from this thesis is the significance of feature selection before method implementation, as an appropriate selection can enhance detection

accuracy [72]. To improve the performance of the methods with potential, future work could focus on training the empirical mean-variance model and RF classifier for longer durations to evaluate whether better results can be achieved. Implementing multi-class classification instead of binary classification might also enhance the performance of the machine learning algorithms, enabling better differentiation between attack traffic and benign traffic.

References

[1] O. Yoachimik and J. Pacheco, *DDoS threat report for 2023 Q1*. [Online]. Available: https://blog.cloudflare.com/ddos-threat-report-2023-q1/ (last visited: May 12, 2023).

[2] M. Antonakakis, T. April, *et al.*, «Understanding the mirai botnet», in *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis.

[3] O. Yoachimik, J. Desgats, and A. Forster, *Cloudflare mitigates record-breaking 71 million request-per-second DDoS attack*. [Online]. Available: https://blog.cloudflare.com/cloudflare-mitigates-record-breaking-71-million-request-per-second-ddos-attack/ (last visited: May 12, 2023).

[4] U. Cisco, «Cisco annual internet report (2018–2023) white paper», *Cisco: San Jose, CA, USA*, vol. 10, no. 1, pp. 1–35, 2020.

[5] Technology Review, *The first DDoS attack was 20 years ago. This is what we've learned since.* [Online]. Available: https://www.technologyreview.com/2019/04/18/103186/the-first-ddos-attack-was-20-years-ago-this-is-what-weve-learned-since/ (last visited: May 16, 2023).

[6] NRK, *Russisk hackergruppe skal ha startet angrep mot norge*. [Online]. Available: https://www.nrk.no/norge/russisk-hackergruppe-skal-ha-startet-angrep-mot-norge-1.16020947 (last visited: May 16, 2023).

[7] Cloudflare, *Application layer DDoS attack*. [Online]. Available: https://www.cloudflare.com/learning/ddos/application-layer-ddos-attack/ (last visited: May 12, 2023).

[8] Z. Liu, C. Hu, and C. Shan, «Riemannian manifold on stream data: Fourier transform and entropy-based DDoS attacks detection method», *Computers & Security*, vol. 109, p. 102 392, 2021.

[9] F. Ullah and M. A. Babar, «Architectural Tactics for Big Data Cybersecurity Analytics Systems: A Review», *Journal of Systems and Software*, vol. 151, pp. 81–118, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121219300172.

[10] H. Wang, X. Liu, *et al.*, «Network security situation awareness based on heterogeneous multi-sensor data fusion and neural network», in *Second International Multi-Symposiums on Computer and Computational Sciences (IMSCCS 2007)*, 2007, pp. 352–359.

[11] G. Gu, A. A. Cárdenas, and W. Lee, «Principled reasoning and practical applications of alert fusion in intrusion detection systems», in *Proceedings of the 2008 ACM symposium on Information, computer and communications security*, 2008, pp. 136–147.

[12] H. T. Elshoush and I. M. Osman, «Alert correlation in collaborative intelligent intrusion detection systems—A survey», *Applied Soft Computing*, vol. 11, no. 7, pp. 4349–4365, 2011, Soft Computing for Information System Security. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156849461000311X.

[13] R. Sadoddin and A. Ghorbani, «Alert correlation survey: Framework and techniques», in *Proceedings of the 2006 international conference on privacy, security and trust: bridge the gap between PST technologies and business services*, 2006, pp. 1–10.

[14] S. T. Zargar, J. Joshi, and D. Tipper, «A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks», *IEEE Communications Surveys & Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[15] J. Mirkovic and P. Reiher, «A Taxonomy of DDoS Attack and DDoS Defense Mechanisms», *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004. [Online]. Available: https://doi.org/10.1145/997150.997156.

[16] NetScout, *Volumetric DDoS Attacks*. [Online]. Available: https://www.netscout.com/what-is-ddos/volumetric-attacks (last visited: Feb. 13, 2023).

[17] NetScout, *ICMP Flood DDoS Attacks*. [Online]. Available: https://www.netscout.com/what-is-ddos/icmp-flood (last visited: Feb. 13, 2023).

[18] NetScout, *UDP Flood Attacks*. [Online]. Available: https://www.netscout.com/what-is-ddos/udp-flood (last visited: Feb. 13, 2023).

[19] K. B. Jørgensen and L. Hansson, *Blacknurse attack*. [Online]. Available: http://blacknurse.dk/ (last visited: May 2, 2023).

[20] NetScout, *Low and Slow DDoS Attacks*. [Online]. Available: https://www.netscout.com/what-is-ddos/low-slow-attack (last visited: May 2, 2023).

[21] L. Zhou, Y. Zhu, *et al.*, «A novel feature-based framework enabling multi-type DDoS attacks detection», *World Wide Web*, pp. 1–23, 2022.

[22] NetScout, *Slowloris DDoS Attacks*. [Online]. Available: https://www.netscout.com/what-is-ddos/slowloris-attacks (last visited: May 2, 2023).

[23] T. A. S. Foundation, *Apache module mod_reqtimeout*. [Online]. Available: https://httpd.apache.org/docs/2.4/mod/mod_reqtimeout.html (last visited: May 3, 2023).

[24] WebTechSurvey, *Apache version usage*. [Online]. Available: https://webtechsurvey.com/technology/apache/versions/ (last visited: May 2, 2023).

[25] Shodan, *Facet Analysis of Apache httpd*. [Online]. Available: https://www.shodan.io/search/facet?query=product%5C%3A%5C%22Apache+httpd%5C%22&facet=version (last visited: May 2, 2023).

[26] NetScout, *Slow Read DDoS Attacks*. [Online]. Available: https://www.netscout.com/what-is-ddos/slow-read-attacks (last visited: May 3, 2023).

[27] B. Stone-Gross, M. Cova, *et al.*, «Your botnet is my botnet: Analysis of a botnet takeover», in *Proceedings of the 16th ACM conference on Computer and communications security*, 2009, pp. 635–647.

[28] K. Linux, *hping3 documentation*. [Online]. Available: https://www.kali.org/tools/hping3/ (last visited: May 2, 2023).

[29] K. Linux, *slowhttptest documentation*. [Online]. Available: https://www.kali.org/tools/slowhttptest/#tool-documentation (last visited: May 2, 2023).

[30] S. Hajj, R. El Sibai, *et al.*, «Anomaly-based intrusion detection systems: The requirements, methods, measurements, and datasets», *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 4, e4240, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4240.

[31] R. A. Disha and S. Waheed, «Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique», *Cybersecurity*, vol. 5, no. 1, p. 1, 2022.

[32] K. Singh, A. Sandu, *et al.*, «Information Theoretic Metrics to Characterize Observations in Variational Data Assimilation», *Procedia Computer Science*, vol. 9, pp. 1047–1055, 2012, Proceedings of the International Conference on Computational Science, ICCS 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050912002347.

[33] C. E. Shannon, «A mathematical theory of communication», *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[34] Y. Xiang, K. Li, and W. Zhou, «Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics», *Trans. Info. For. Sec.*, vol. 6, no. 2, pp. 426–437, Jun. 2011. [Online]. Available: https://doi.org/10.1109/TIFS.2011.2107320.

[35] IBM, *What is machine learning?* [Online]. Available: https://www.ibm.com/topics/machine-learning (last visited: May 4, 2023).

[36] T. Yiu, *Understanding Random Forest*. [Online]. Available: https://towardsdatascience.com/understanding-random-forest-58381e0602d2 (last visited: May 4, 2023).

[37] E. E. (LEDU), *Understanding K-means Clustering in Machine Learning*. [Online]. Available: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1 (last visited: May 6, 2023).

[38] C. Hua and Z. Kai, «Multisensor data fusion for new generation distributed intrusion detection systems of ship», in *2011 International Conference on E-Business and E-Government (ICEE)*, 2011, pp. 1–4.

[39] T. Zang, X. Yun, and Y. Zhang, «A survey of alert fusion techniques for security incident», in *2008 The Ninth International Conference on Web-Age Information Management*, 2008, pp. 475–481.

[40] D. Xu and P. Ning, «Correlation Analysis of Intrusion Alerts», in *Intrusion Detection Systems*. Boston, MA: Springer US, 2008, pp. 65–92. [Online]. Available: https://doi.org/10.1007/978-0-387-77265-3_4.

[41] IBM, *What is NetFlow?* [Online]. Available: https://www.ibm.com/topics/netflow (last visited: May 7, 2023).

[42] B. Claise, J. Clarke, and J. Lindblad, «Network Programmability with YANG», in Addison-Wesley Professional, 2019, pp. 33–37.

[43] B. Claise, *Cisco Systems NetFlow Services Export Version 9*, RFC 3954, Oct. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3954.

[44] R. Hofstede, P. Čeleda, *et al.*, «Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX», *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[45] L. D. Tsobdjou, S. Pierre, and A. Quintero, «An Online Entropy-Based DDoS Flooding Attack Detection System With Dynamic Threshold», *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1679–1689, 2022.

[46] L. Feinstein, D. Schnackenberg, *et al.*, «Statistical approaches to DDoS attack detection and response», in *Proceedings DARPA Information Survivability Conference and Exposition*, vol. 1, 2003, 303–314 vol.1.

[47] V. Sekar, N. G. Duffield, *et al.*, «LADS: Large-scale Automated DDoS Detection System.», in *USENIX Annual Technical Conference, General Track*, 2006, pp. 171–184.

[48] J. David and C. Thomas, «Efficient DDoS Flood Attack Detection Using Dynamic Thresholding on Flow-Based Network Traffic», *Comput. Secur.*, vol. 82, no. C, pp. 284–295, May 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2019.01.002.

[49] J. H. Corrêa, P. M. Ciarelli, *et al.*, «ML-Based DDoS Detection and Identification Using Native Cloud Telemetry Macroscopic Monitoring», *Journal of Network and Systems Management*, vol. 29, pp. 1–28, 2021.

[50] M. A. S. Monge, A. H. González, *et al.*, «Traffic-flow analysis for source-side DDoS recognition on 5G environments», *Journal of Network and Computer Applications*, vol. 136, pp. 114–131, 2019.

[51] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, «SDN-Based Architecture for Transport and Application Layer DDoS Attack Detection by Using Machine and Deep Learning», *IEEE Access*, vol. 9, pp. 108 495–108 512, 2021.

[52] T. A. Tuan, H. V. Long, *et al.*, «Performance evaluation of Botnet DDoS attack detection using machine learning», *Evolutionary Intelligence*, vol. 13, pp. 283–294, 2020.

[53] X. Liang and T. Znati, «On the performance of intelligent techniques for intensive and stealthy DDos detection», *Computer Networks*, vol. 164, p. 106 906, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128619302968.

[54] J. Mazel, P. Casas, *et al.*, «Hunting attacks in the dark: clustering and correlation analysis for unsupervised anomaly detection», *International Journal of Network Management*, vol. 25, no. 5, pp. 283–305, 2015. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.1903.

[55] DFN-CERT, *NeMo / DDoS Application: User manual*. [Online]. Available: https://security.geant.org/wp-content/uploads/2022/12/NeMo-User_Manual-20200513.pdf (last visited: Jun. 29, 2023).

[56] F. Maggi, M. Matteucci, and S. Zanero, «Reducing false positives in anomaly detectors through fuzzy alert aggregation», *Information Fusion*, vol. 10, no. 4, pp. 300–311, 2009, Special Issue on Information Fusion in Computer Security. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156625350900013X.

[57] Y. Yao, Z. Wang, *et al.*, «Multi-source alert data understanding for security semantic discovery based on rough set theory», *Neurocomputing*, vol. 208, pp. 39–45, 2016, SI: BridgingSemantic. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231216304817.

[58] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, «An alert analysis approach to DDoS attack detection», in *2016 International Conference on Accessibility to Digital World (ICADW)*, IEEE, 2016, pp. 33–38.

[59] R. J. Wieringa, *Design science methodology for information systems and software engineering*. Springer, 2014.

[60] L. F. Gustavsen, «DDoS detection using multi-source analysis», Department of Information Security, Communication Technology, NTNU – Norwegian University of Science, and Technology, Project report in TTM4502, Dec. 2022.

[61] S. Petrovic, G. Alvarez, *et al.*, «Labelling Clusters in an Intrusion Detection System Using a Combination of Clustering Evaluation Techniques», in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences (HICSS'06)*, vol. 6, 2006, 129b–129b.

[62] C. Nepal, *Did you know about the Xmas attack?* [Online]. Available: https://medium.com/cryptogennepal/did-you-know-about-the-xmas-attack-c6daae776a0e (last visited: May 18, 2023).

[63] P. Kumar, M. Tripathi, *et al.*, «SAFETY: Early Detection and Mitigation of TCP SYN Flood Utilizing Entropy in SDN», *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1545–1559, 2018.

[64] A. Abdelkefi, Y. Jiang, and X. Dimitropoulos, «K-sparse Approximation for Traffic Histogram Dimensionality Reduction», in *2012 8th international conference on network and service management (cnsm) and 2012 workshop on systems virtualiztion management (svm)*, 2012, pp. 64–72.

[65] D. Merkel, «Docker: Lightweight linux containers for consistent development and deployment», *Linux J.*, vol. 2014, no. 239, Mar. 2014.

[66] The IEEE and The Open Group, *crontab*. [Online]. Available: https://pubs.opengroup.org/onlinepubs/007904975/utilities/crontab.html (last visited: Jul. 3, 2023).

[67] The pandas development team, *Pandas-dev/pandas: Pandas*, version v1.5.3, Jan. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7549438.

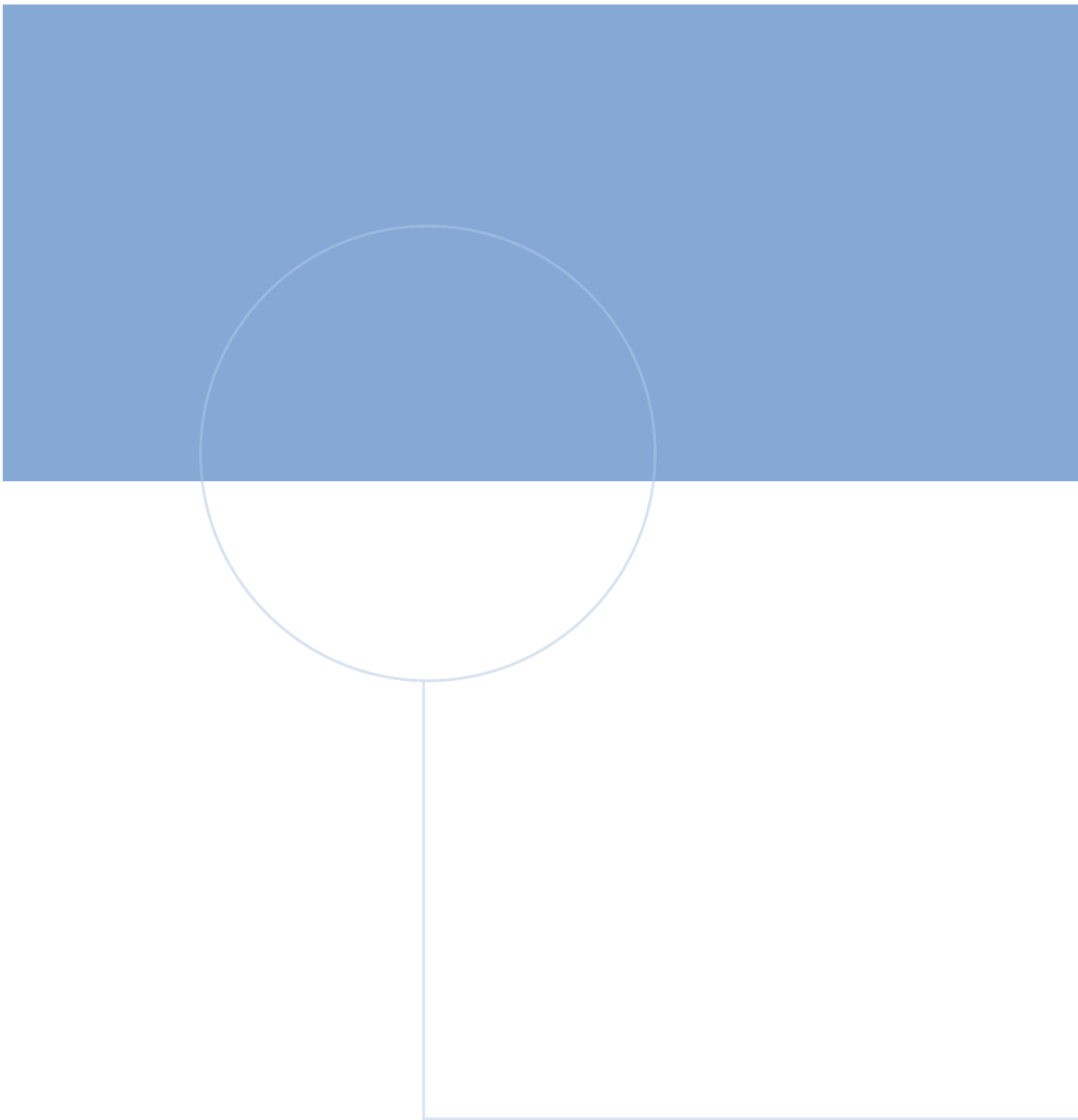
[68] C. R. Harris, K. J. Millman, *et al.*, «Array programming with NumPy», *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2.

[69] F. Pedregosa, G. Varoquaux, *et al.*, «Scikit-learn: Machine learning in Python», *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[70] E. Foundation, *Eclipse Mosquitto™*. [Online]. Available: https://mosquitto.org/ (last visited: May 21, 2023).

[71] N. developers, *NetworkX: Network Analysis in Python*. [Online]. Available: https://networkx.org/ (last visited: May 1, 2023).

[72] R. Zuech, T. M. Khoshgoftaar, and R. Wald, «Intrusion detection and big heterogeneous data: A survey», *Journal of Big Data*, vol. 2, no. 1, pp. 1–41, 2015.



NTNU

Norwegian University of
Science and Technology