



**BRNO UNIVERSITY OF TECHNOLOGY**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

**FACULTY OF INFORMATION TECHNOLOGY**

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

**DEPARTMENT OF COMPUTER SYSTEMS**

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

**IDENTIFICATION OF DOS PACKETS**

IDENTIFIKACE DOS PAKETŮ

**BACHELOR'S THESIS**

BAKALÁŘSKÁ PRÁCE

**AUTHOR**

AUTOR PRÁCE

**MARIÁN VALKÓ**

**SUPERVISOR**

VEDOUCÍ PRÁCE

**Ing. MARTIN ŽÁDNÍK, Ph.D.**

**BRNO 2025**

# Bachelor's Thesis Assignment



161527

Institut: Department of Computer Systems (DCSY)  
Student: **Valkó Marián**  
Programme: Information Technology  
Title: **Identification of DoS packets**  
Category: Networking  
Academic year: 2024/25

## Assignment:

1. Survey the existing literature about DoS attacks and their types. Focus on application-level attacks.
2. Survey the existing literature about the techniques of DoS detection using heuristics including machine learning.
3. Design a method for DoS detection that would focus on a specific class of DoS attacks.
4. Implement the proposed method.
5. Evaluate the implementation in a laboratory environment.
6. Discuss the outcomes and futures improvements.

## Literature:

Dle pokynů vedoucího.

Requirements for the semestral defence:

Fulfilling the first to the third item of the assignment.

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

Supervisor: **Žádník Martin, Ing., Ph.D.**  
Head of Department: Sekanina Lukáš, prof. Ing., Ph.D.  
Beginning of work: 1.11.2024  
Submission deadline: 14.5.2025  
Approval date: 31.10.2024

## Abstract

As online services become essential to everyday life, ensuring their availability is increasingly critical. Distributed denial-of-service attacks are one of the many threats to this availability. This thesis proposes an offline machine learning-based approach for detecting such attacks at the application layer, using an unsupervised technique to learn normal traffic patterns and identify anomalies. The proposed approach shows promising results and reveals both the strengths and limitations of this method.

## Abstrakt

Keďže online služby sa stávajú neoddeliteľnou súčasťou každodenného života, zabezpečenie ich dostupnosti je čoraz dôležitejšie. Distributed denial-of-service útoky predstavujú jednu z mnohých hrozieb pre túto dostupnosť. Táto práca navrhuje offline prístup založený na strojovom učení na detekciu takýchto útokov na aplikačnej vrstve, pričom využíva učenie bez učiteľa na osvojenie si charakteristík bežnej prevádzky a identifikáciu anomálií. Navrhovaný prístup priniesol sľubné výsledky a odhalil silné stránky aj obmedzenia tejto metódy.

## Keywords

DDoS, anomaly detection, application-layer attacks, unsupervised learning, autoencoder, LSTM, network traffic, flow-based features, reconstruction error, detection threshold

## Klíčová slova

DDoS, detekcia anomálií, útoky na aplikačnej vrstve, učenie bez učiteľa, autoenkóder, LSTM, sieťová prevádzka, charakteristiky tokov, rekonštrukčná chyba, detekčný prah

## Reference

VALKÓ, Marián. *Identification of DoS packets*. Brno, 2025. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Martin Žád- ník, Ph.D.

## Rozšírený abstrakt

V dnešnej digitálnej dobe sa webové aplikácie a sieťové služby stali neoddeliteľnou súčasťou nášho každodenného života. Umožňujú nám komunikovať, pristupovať k vládny portálom, vyhľadávať informácie, spravovať financie a pohodlne nakupovať z domova. S rastúcim počtom online služieb sa zvyšuje aj potreba zabezpečenej a spoľahlivej sieťovej infraštruktúry. Výpadky služieb môžu spôsobiť finančné straty, pokles produktivity a niekedy aj znemožniť prístup ku kritickým systémom.

Medzi najstaršie a pretrvávajúce hrozby pre online systémy patria útoky typu Denial of Service (DoS). Tieto útoky zahlcujú cieľové zariadenie takým množstvom požiadaviek, že prestane reagovať na legitímnu prevádzku. S nárastom distribuovaných DoS (DDoS) útokov vykonávaných prostredníctvom sietí napadnutých zariadení (botnetov) sa ich dopad výrazne zvýšil. Postupne, ako sa zdokonaľovali obranné mechanizmy, útočníci presunuli pozornosť na sofistikovanejšie útoky mierené na aplikačnú vrstvu.

DDoS útoky na aplikačnej vrstve predstavujú mimoriadnu výzvu, pretože často vytvárajú prevádzku, ktorá sa navonok javí ako legitímna, keďže napodobňuje bežné interakcie používateľov so servermi či API. Namiesto zahltenia šírky pásma útočníci mieria na vyčerpanie konkrétnych serverových zdrojov ako pamäť, procesorový čas alebo databázové spojenia. Takéto útoky sa ťažšie odhaľujú a ich identifikácia vyžaduje hlbšiu analýzu sieťových charakteristík.

Táto práca sa zameriava na detekciu aplikačných DDoS útokov pomocou analýzy anomálií založenej na rekonštrukčnej chybe. Navrhovaný systém využíva autoenkodér založený na Long Short-Term Memory (LSTM) sieťach, trénovaný metódou učenia bez učiteľa. Model sa učí rozpoznávať normálne charakteristiky sieťovej prevádzky a následne identifikuje odchýlky, ktoré môžu naznačovať útok.

Systém spracováva sieťovú prevádzku z PCAP súborov, extrahuje 19 kľúčových charakteristík každého sieťového toku a vytvára sekvencie zachytávajúce ich vývoj v čase. Tieto sekvencie sa normalizujú a spracovávajú LSTM autoenkodérom, ktorý sa učí rekonštruovať len normálnu prevádzku. Pri detekcii sa ako indikátor anomálií používa rekonštrukčná chyba, kde vyššia hodnota signalizuje potenciálne škodlivé správanie.

Experimenty na dátovej sade CICIDS2017 ukazujú, že navrhovaný prístup dokáže úspešne identifikovať viaceré typy DDoS útokov vrátane DoS Hulk, GoldenEye, Slowloris a Slowhttptest. Najlepšie výsledky sa dosiahli pri použití časového okna s dĺžkou 0,5 sekundy a pri nastavení prahovej hodnoty pomocou Gaussovského zmiešaného modelu, pričom F1 skóre dosiahlo 94,29%. Ďalšie experimenty analyzovali vplyv rôznych dĺžok časových okien, alternatívnych metód nastavovania prahu a tiež výber a redukciu charakteristík popisujúcich sieťovú prevádzku.

Testovanie na reálnej internetovej prevádzke z dátovej sady MAWI potvrdilo, že systém dokáže dobre generalizovať v rámci rovnakého sieťového prostredia a udržať nízku mieru falošných poplachov. Na druhej strane, aplikovanie modelu na úplne odlišnú dátovú sadu prinieslo výrazne slabšie výsledky, čo naznačuje, že model by sa mal trénovať priamo na prevádzke z prostredia, v ktorom bude detekcia v praxi prebiehať.

Napriek určitým obmedzeniam navrhovaná metóda úspešne deteguje útoky na aplikačnej vrstve a predstavuje sľubný prístup k ochrane sieťových služieb.

# Identification of DoS packets

## Declaration

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

.....  
Marián Valkó  
May 12, 2025

## Acknowledgements

I would like to express my sincere gratitude to my supervisor, Ing. Martin Žádník, Ph.D., for his continuous support, guidance, and valuable insights throughout the development of this thesis. His expertise, encouragement, and constructive feedback have been instrumental in shaping the direction and quality of my work.

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Background</b>	<b>3</b>
2.1	Evolution of DDoS attacks . . . . .	3
2.2	Architectures of DDoS attacks . . . . .	4
2.3	Classification of DDoS attacks . . . . .	6
2.4	Application layer attacks . . . . .	7
<b>3</b>	<b>Related work</b>	<b>10</b>
3.1	Statistical techniques . . . . .	10
3.2	Machine learning techniques . . . . .	12
3.3	LSTM-Autoencoder for anomaly-based DDoS detection . . . . .	14
<b>4</b>	<b>The proposed approach and implementation</b>	<b>16</b>
4.1	System overview and workflow . . . . .	16
4.2	Preprocessing and feature extraction . . . . .	18
4.3	Sequence creation and temporal pattern modeling . . . . .	19
4.4	Model architecture and training . . . . .	21
4.5	Detection and threshold determination . . . . .	24
<b>5</b>	<b>Experiments and evaluation</b>	<b>25</b>
5.1	Datasets . . . . .	25
5.2	Evaluation metrics and methodology . . . . .	27
5.3	Impact of detection thresholds and model reproducibility . . . . .	28
5.4	Comparison of threshold method performance . . . . .	30
5.5	Time window size effects . . . . .	31
5.6	Feature reduction impact on detection performance . . . . .	33
5.7	Evaluation using the MAWI dataset . . . . .	34
5.8	Cross-Dataset generalization . . . . .	35
<b>6</b>	<b>Lessons learned and future work</b>	<b>36</b>
<b>7</b>	<b>Conclusion</b>	<b>38</b>
	<b>Bibliography</b>	<b>39</b>

# Chapter 1

## Introduction

In today’s digital world, the use of web applications and network services has become part of our everyday life. They allow us to communicate, access government portals, search for information, manage finances, and conveniently shop from our homes. As more organizations and services move online, the need for secure and reliable network infrastructure becomes increasingly important. Disruptions can result in financial loss, productivity setbacks, and even denial of access to critical services.

Among the threats targeting online systems, Denial of Service attacks remain one of the oldest and most persistent. These attacks typically involve overwhelming a server with requests from a single source until it can no longer respond to legitimate traffic. Their impact became even more severe with the rise of distributed denial-of-service attacks, where attackers use entire networks of compromised systems, so-called botnets, to flood a target from many sources at once. Over time, as defenses against volumetric and protocol-level attacks improved, attackers began shifting their focus toward more sophisticated techniques, especially those targeting the application layer.

Application-layer DDoS attacks present a unique challenge. They often generate traffic that appears legitimate on the surface, mimicking normal user interactions with web servers or APIs. Instead of overwhelming bandwidth, they aim to exhaust specific server resources such as memory, CPU, or database connections. This makes them more challenging to detect, as their traffic patterns often fall within the range of what is considered normal.

Over the years, many approaches have been proposed to address the challenge of detecting Distributed Denial of Service attacks, ranging from statistical analysis of traffic patterns to machine learning techniques designed to identify anomalies. This thesis explores the use of temporal modeling and reconstruction-based anomaly detection to distinguish between normal and malicious traffic patterns, with a focus on application-layer DDoS scenarios.

Chapter 2 explores the background and evolution of DDoS attacks, their architectures, classification, and the specific challenges associated with the application layer. Chapter 3 surveys existing detection techniques, including statistical methods and machine learning approaches, and discusses how they have been applied to detect various forms of DDoS traffic. The core detection system developed in this thesis is described in chapter 4, covering its architectural design, data preprocessing, temporal modeling, and detection strategy. Chapter 5 presents the evaluation setup and experimental results, highlighting the system’s performance. Finally, chapter 6 summarizes lessons learned throughout the development process and outlines potential directions for further research, and chapter 7 concludes the thesis.

# Chapter 2

## Background

This chapter focuses on Distributed Denial of Service attacks and their evolution into significant cybersecurity threats. Section 2.1 examines the historical development and increasing sophistication of DDoS attacks. Section 2.2 analyzes DDoS attack architectures, and Section 2.3 discusses the classification of DDoS attacks. Finally, section 2.4 provides a closer look into application layer DDoS attack.

### 2.1 Evolution of DDoS attacks

As discussed in the survey by Praseed and Thilagam [23], earlier DDoS attacks were straightforward, attackers would flood networks with massive amounts of traffic using protocols like UDP and ICMP, effectively overwhelming the network's bandwidth. However, as networks and servers became more robust, attackers had to change their approach. This led to a shift towards transport layer attacks, which also became easier to detect because of their predictable patterns. The most recent evolution has been towards application layer DDoS attacks, which, instead of flooding the network, target server resources like CPU, memory, and database connections, making them both more effective and harder to identify.

As highlighted by A10 Networks [3], the DDoS threat landscape has continued to evolve, with the number of attacks increasing both in scale and sophistication. Events such as the COVID-19 lockdown, the war in Ukraine, and the emergence of new attack vectors such as the HTTP/2 Rapid Reset have demonstrated the growing impact of DDoS attacks. The rise of carpet bombing techniques and the increasing use of complex botnets further complicate detection and mitigation efforts. Notable recent incidents include record-setting attacks on AWS and Google, where peak volumes reached 2.3 Tbps and 2.5 Tbps, respectively, highlighting the scale modern DDoS attacks can achieve.

A recent comprehensive study of DDoS evolution by Wang et al. [31] emphasizes a significant trend that modern DDoS campaigns increasingly leverage a wide range of network protocols and systems, from HTTP/2 and QUIC to IoT-specific services and blockchain infrastructure. Attackers have become more strategic, utilizing stealthy, low-rate traffic patterns and encrypted payloads to evade conventional detection systems. Additionally, the growing number of vulnerable cloud services and exposed APIs continues to expand the attack surface, allowing adversaries to exploit poorly secured endpoints and amplify the intensity of the attack.

Similarly, Akamai’s industry report [4] identifies a sharp resurgence in politically motivated DDoS campaigns. Hactivist groups such as Killnet have increasingly targeted financial institutions, government services, and critical infrastructure with the goal of disrupting rather than financial gain. The widespread availability of DDoS-as-a-service platforms has significantly reduced the technical barrier to launching attacks. Furthermore, attackers are now combining volumetric assaults with extortion demands or using DDoS as a cover to hide more invasive operations like data theft or espionage.

The landscape has also matured technically. In 2010, the top five attack vectors represented 90% of all DDoS traffic. By 2022, this share decreased to 55%, indicating that a larger and more sophisticated array of vectors is being used. Attackers are focused not only on overwhelming bandwidth but also on targeting DNS services, web servers, and even remote access infrastructure, a critical risk in an era of widespread remote work.

Table 2.1 summarizes selected notable DDoS attacks based on the findings reported by A10 Networks [3].

<b>Attack</b>	<b>Description</b>	<b>Peak Volume</b>
HTTP/2 Rapid Reset Attack (2023)	Exploited the HTTP/2 protocol’s RST_STREAM feature, overwhelming servers by rapidly opening and resetting streams.	Not provided, stated as record breaking
Google UDP Amplification Attack (2020)	Used spoofed UDP traffic via CLDAP, DNS, and SMTP servers; sustained over six months.	2.5 Tbps
AWS CLDAP Reflection Attack (2020)	Targeted AWS customer by amplifying traffic through vulnerable CLDAP servers.	2.3 Tbps
Mirai Botnet Attacks (2016)	Used IoT devices to launch massive attacks, including the Dyn DNS outage affecting major platforms like Twitter and Netflix.	1.1–1.5 Tbps
GitHub Memcached Attack (2018)	Leveraged Memcached servers for massive amplification, briefly taking GitHub offline.	1.35 Tbps

Table 2.1: Summary of notable DDoS attacks and their description.

## 2.2 Architectures of DDoS attacks

The difference between a Denial of Service attack and a Distributed Denial of Service attack mainly differs in scale. A Denial of Service attack by itself aims to overload the target server or network to an extent where legitimate traffic can no longer be processed. Today, this type of attack is generally considered less effective and is unlikely to cause significant disruption on its own.

A Distributed Denial of Service attack shares the same goal, nevertheless in a much more effective way, as it utilizes numerous compromised internet-connected systems infected with malware, making it easier for the attackers to control them remotely. These infected devices are often called bots, where a number of bots create a botnet. After an attacker establishes

a botnet, they can send remote commands to each bot and direct the attack. Each bot in a botnet then sends requests to the targeted IP addresses in an attempt to disrupt or ultimately make the service unavailable as highlighted by Adedeji et al [5].

The architecture of a DDoS attack consists of the target system, an attacker, and a botnet. However, Huang et al. [15] classify the architectures into two types based on the management mode of a botnet, namely centralized and decentralized.

In a centralized DDoS attack architecture, all the machines are connected to a system of command-and-control (C&C) servers, and the botnet is controlled by directly sending commands to the bots. The participants in such a DDoS attack architecture are the attacker, a command-and-control system, a botnet, and the targeted server. The advantages of such an architecture are that the botnet is undetectable by monitoring the communication between bots, and the botnet's real-time management enables threat actors to easily adjust the attack strategy. A downside is its weak robustness, if a C&C system fails, it leads to the breakdown of the whole botnet. Figure 2.1 shows a diagram of the centralized attack architecture.

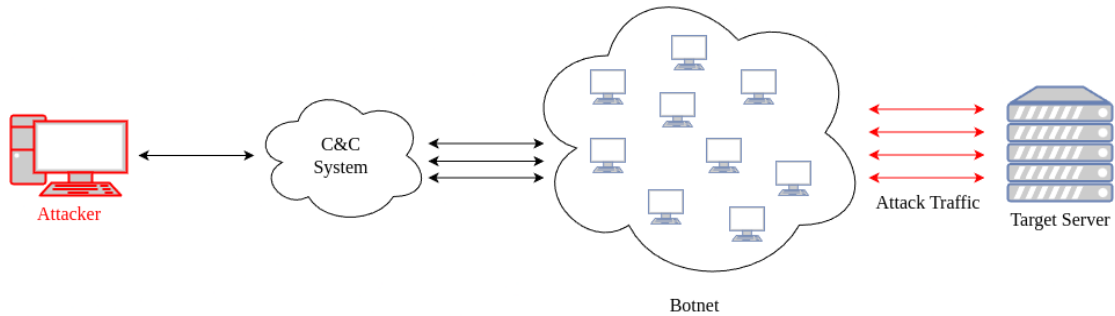


Figure 2.1: Centralized DDoS attack architecture

In a decentralized DDoS attack, on the other hand, the bots form a peer-to-peer (P2P) network. This approach became famous for large botnets as it enables the attacker to indirectly manage the botnet by issuing instructions to a single machine, which then forwards them to all other machines through the P2P network. This architecture made the botnet more robust. However, a downside is that it may be easier to detect by monitoring the communication of the P2P network. Figure 2.2 shows a diagram of the decentralized attack architecture.

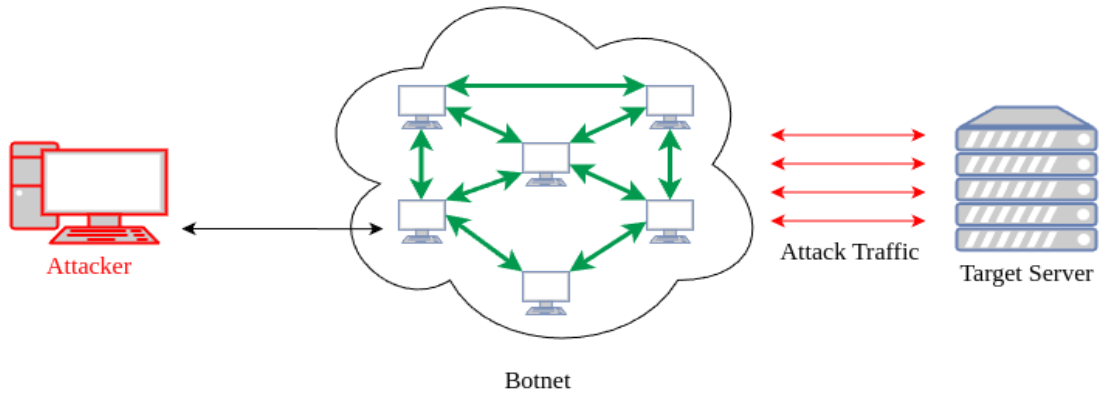


Figure 2.2: Decentralized DDoS attack architecture

### 2.3 Classification of DDoS attacks

Although DDoS attacks can target any of the seven layers of the OSI model, they are most often divided into three broad categories based on which layer they exploit, as identified by Merkebauly [20]:

- **Volume-Based Attacks.** These attacks are one of the most common and aim to flood a target's bandwidth, making services unavailable. Attackers send huge volumes of data to overwhelm the system. They typically focus on Layer 3 (Network Layer) and Layer 4 (Transport Layer), preventing regular traffic from getting through.
- **Protocol Attacks.** These take advantage of weaknesses in communication protocols to drain server resources or disrupt important services. They send a high volume of packets each second, targeting vulnerabilities in how protocols function. Like volume attacks, these also focus on Layer 3 and Layer 4 of the OSI model, often exploiting issues in the handshake or session setup.
- **Application Layer Attacks.** Also known as Layer 7 attacks, they focus on the highest level of the OSI model. Instead of attacking the infrastructure directly, they aim to exhaust the resources of a specific application or service. These attacks can be tricky to spot because they often mimic legitimate user traffic and requests.

Table 2.2 summarizes the key characteristics of the most known DDoS attacks, including the targeted OSI layer, attack rate, affected resources, and a brief description.

Attack Type	OSI Layer	Attack Rate	Targeted Resource	Description
UDP Flood	Layer 4	High (Bps/Gbps)	Network bandwidth	Overwhelms the target system by sending large volumes of UDP packets, consuming available bandwidth and preventing legitimate service access.
ICMP (Ping) Flood	Layer 3	High (Bps/Gbps)	Network bandwidth	Sends excessive ICMP echo requests to consume network bandwidth, resulting in degraded network performance.
TCP SYN Flood	Layer 4	High (Pps)	Server resources (CPU, memory)	Exploits TCP handshake by creating numerous incomplete connections, exhausting server resources and connection tables.
Ping of Death	Layer 3	Low	System stability	Sends malformed or oversized packets that exceed normal size limits, potentially causing system crashes or instability.
Smurf Attack	Layer 3	High (Pps)	Network bandwidth	Utilizes IP spoofing and broadcast addresses to generate amplified ICMP traffic, directing multiple responses toward the target.
HTTP GET Flood	Layer 7	High (Rps)	Application resources	Generates high volumes of HTTP GET requests to exhaust server resources and connection capacity.
HTTP POST Flood	Layer 7	High (Rps)	Application resources	Sends multiple HTTP POST requests with large payloads, consuming server processing power and memory.
Slowloris	Layer 7	Low, persistent connections	Server threads	Creates multiple incomplete HTTP connections that remain open, depleting the server's connection pool and blocking new connections.

Table 2.2: Key characteristics of common DDoS attacks categorized by OSI layer.

## 2.4 Application layer attacks

Application layer DDoS attacks have several distinct characteristics as described by Praseed et al. [23] that set them apart from traditional DDoS attacks. First, these attacks use

completely legitimate HTTP requests without any apparent difference between an attack request and a normal user request except for the attacker’s intent. This makes traditional packet filters and many firewalls struggle to detect them effectively. Unlike network layer attacks that need massive traffic volumes to overwhelm bandwidth, application layer attacks can bring down a server with fewer requests by targeting specific server resources. This lower traffic volume means that many traditional detection systems, which rely on identifying traffic spikes, have become ineffective. Another key feature is their precision. Attackers can specifically target different server components like CPU, database, memory, or socket connections. An attack focusing on one resource might barely affect others but can still crash the entire system. This means that protecting one resource does not necessarily help protect others. Probably the most challenging aspect is the similarity of these attacks to flash crowds, which are sudden spikes in legitimate traffic that usually take place during major events or sales. These situations create an increase in the number of legitimate HTTP requests, making it crucial but difficult to distinguish between them. This distinction is crucial since blocking legitimate flash crowd traffic could significantly affect a website’s business.

Tripathi and Hubballi [29] further classify application layer attacks into two broad categories, protocol-specific and generic. Protocol-specific attacks focus on vulnerabilities or expected behavior in a particular application-layer protocol such as HTTP, DNS, or SIP. Generic attacks, on the other hand, exploit connection-level features shared across many protocols, for example, holding connections open or sending traffic just slowly enough to evade timeouts. This classification highlights how these attacks are not limited to a single protocol and can often be ported across services with minimal changes.

A growing concern is the exploitation of newer protocol features. For example, while HTTP/2 improves performance through multiplexing, it also introduces opportunities for abuse. Attacks that delay or omit HTTP message bodies, similar to Slowloris, can still tie up server threads even under HTTP/2. Another technique involves abusing the HTTP PRAGMA header to reset timeouts and prolong connections indefinitely.

Stealthy attacks are also evolving. Variants such as SlowReq, SlowDroid, and Slowcomm send partial or minimal data over long durations. These attacks work against any service that supports persistent TCP connections, including web, mail, and file transfer servers. Their low and consistent traffic volume makes them hard to detect while still exhausting memory, threads, or sockets on the target server.

Real-world incidents illustrate the impact of these techniques. Tripathi and Hubballi [29] mention that an Imperva customer was hit with an HTTP flood peaking at 292,000 requests per second. Other documented cases include DNS-based application attacks that disrupted GitHub and HSBC, and HTTP floods that impacted the Rio Olympics.

Beyond the classification from Tripathi and Hubballi, Mantas et al. [19] provide further insight into advanced application-layer DoS strategies. These include so-called „asymmetric“ or „one-shot“ attacks that can exhaust server resources using just a few requests. For example, HashDoS attacks trigger CPU-intensive hash collisions in web servers, while ReDoS abuses poorly constructed regular expressions to waste server time. XDoS attacks send deeply nested or oversized XML structures, which overwhelm XML parsers and memory.

Some attacks don’t rely on software bugs but instead take advantage of how applications are meant to work. For instance, flooding a site with complex search queries or log-in attempts may not violate any rules but still strain system resources.

Mantas et al. also classify application-layer DDoS attacks according to six key features:

- **Attack target level:** whether the attack hits application logic such as business-process or database operations, or leverages protocol behavior such as HTTP, DNS, or SMTP.
- **Exploitation type:** whether it exploits a specific vulnerability or instead abuses intended features by either flooding the service or crafting individual, heavy-workload requests.
- **Attack methodology:** whether traffic is sent directly from the attacker’s bots or is reflected and amplified via third-party services.
- **Traffic volume:** whether the attacker relies on large bursts of requests or on low-volume patterns such as periodic pulses or very slow connections to tie up resources.
- **Request workload:** whether each request imposes only modest work or is designed to generate high CPU or memory load.
- **Affected services:** whether the attack isolates a single component or inadvertently disrupts multiple services, possibly across layers.

Table 2.3 provides a brief description of selected application-layer attacks and their categorization based on traffic volume.

Attack	Description	Category
HTTP flood (GET/POST)	Sends many HTTP requests targeting backend operations to exhaust server resources.	High-volume
SOAP flooding	Overloads web services by sending excessive SOAP/XML requests.	High-volume
DNS-based application attack	Bombards DNS servers with heavy queries, disrupting resolution.	High-volume
SIP flooding	Sends numerous SIP INVITE or REGISTER requests to exhaust VoIP session resources.	High-volume
Slowloris	Sends HTTP headers slowly to keep connections open, exhausting web server threads.	Low-volume
Slow Message Body	Delays or never finishes HTTP message body, tying up server threads.	Low-volume
Slow Read	Requests large files but uses small TCP window to slow responses.	Low-volume
HTTP PRAGMA Abuse	Uses PRAGMA headers to reset timeout clocks and hold idle connections.	Low-volume
SlowReq / SlowDroid	Sends small data chunks slowly to hold open connections.	Low-volume
HashDoS	Triggers hash collisions to overwhelm CPU usage.	Low-volume
ReDoS	Sends crafted input to trigger expensive regex processing.	Low-volume
XDoS	Sends nested or oversized XML to overload parsers and memory.	Low-volume

Table 2.3: Categorization of application-layer DDoS attacks and their characteristics

# Chapter 3

## Related work

This chapter provides an overview of various techniques used in DDoS detection. Section 3.1 describes statistical approaches for identifying attacks based on traffic behavior and briefly classifies different types of statistical detection methods. Section 3.2 outlines machine learning techniques for DDoS detection and provides a brief classification based on the learning type. Section 3.3 first introduces the individual components, Long Short-Term Memory networks and autoencoders, before reviewing a prior LSTM-Autoencoder-based detection approach from previous research.

### 3.1 Statistical techniques

One approach for detecting Distributed Denial of Service attacks is to analyze the statistical properties of network traffic. In these techniques, a model is built from regular traffic data to define what is considered “normal.” New traffic is then compared against this baseline, and significant deviations in key metrics, such as packet sizes, flow frequencies, or inter-arrival times, may signal a potential attack.

A review by Khalaf et al. [16] divides statistical detection methods into two main categories:

- **Parametric methods:** These methods assume that traffic follows a specific probability distribution and use calculated parameters such as the mean and standard deviation to characterize normal behavior. For example, if normal traffic typically has an average packet size of 500 bytes with limited variation, a persistent deviation from these values may indicate abnormal activity.
- **Non-Parametric methods:** In contrast, nonparametric methods do not assume any particular distribution. Instead, they rely directly on empirical data, using techniques such as histograms or kernel density estimation, to capture and monitor real-time changes in traffic patterns. This flexibility allows these methods to better handle the diverse and dynamic nature of network traffic.

Some of the statistical techniques used in recent research are:

- **Z-Score & Coefficient of variation:** Majed et al. [18] presented a lightweight statistical approach for DDoS detection designed for deployment at the ISP level. Their method aggregates NetFlow statistics and calculates the Z-score for key traffic features, using the mean and standard deviation as benchmarks of normal behavior.

By comparing Z-score values to dynamically estimated thresholds derived from the coefficient of variation, the system identifies outliers that may indicate DDoS attacks. The approach demonstrated nearly 100% detection rates within a 30-second time window, highlighting both its accuracy and low computational overhead. This method is practical for real-time detection and adapts to changing traffic conditions without manual threshold tuning.

- **4EMA:** Bojovic et al. [10] developed a hybrid detection technique that combines exponential moving averages applied to both feature-based and volume-based traffic indicators. Their method uses dual EMA filters, one short term and one long-term, on packet entropy and packet counts to capture significant shifts while filtering out short-term noise. The difference between these EMA values serves as the detection signal, with thresholds used to trigger or clear alarms. Evaluated in a real academic network under both high-rate and low-rate DDoS attack scenarios, the approach achieved 100% detection with minimal false positives and outperformed traditional CUSUM-based detectors in flexibility and reliability.
- **ESPRT:** Ali et al. [6] combined entropy measurement with the Sequential Probability Ratio Test to detect DDoS anomalies. They compute the Shannon entropy of network flows to assess traffic randomness and then apply SPRT to decide if the observed values indicate an attack. This integrated approach overcomes the limitations of fixed threshold methods and helps reduce false positives. Their evaluations on public datasets showed high detection accuracy and strong F-scores.
- **CSR & FDM:** Amma et al. [7] introduced a statistical method that combines feature selection and extraction to detect DoS attacks. Their approach first ranks network traffic features using the class scatter ratio, which measures how effectively each feature distinguishes normal traffic from attack traffic. Then, a feature distance map is constructed to capture the correlations among the selected features. The method builds separate profile vectors for normal and attack traffic and classifies new traffic by comparing its FDM to these profiles. Evaluations demonstrated high detection accuracy with low false positive rates and efficient computation, making it suitable for real-time attack detection.
- **MCA:** Tan et al. [28] proposed a system that employs multivariate correlation analysis for DoS attack detection. Their approach extracts basic traffic features and uses a triangle-area based technique to capture the geometric correlations between pairs of features, resulting in a triangle area map for each traffic record. These TAMs are then used to generate normal and attack profile vectors by computing the Mahalanobis distance between new traffic records and the established normal profile. The method effectively classifies traffic as legitimate or as an attack by comparing these distances against a predefined threshold. Evaluations on benchmark datasets, such as KDD Cup 99, showed that this MCA-based approach achieves high detection accuracy with a low false positive rate.
- **KDE:** Hu et al. [14] introduced an anomaly detection method based on local kernel density estimation using a novel Volcano kernel. Their approach calculates each sample's anomaly factor by comparing its local density to the weighted density of its neighbors. The Volcano kernel ensures that normal samples within clusters receive anomaly factors close to 1, while anomalies receive significantly higher values. A

weighted neighborhood density improves robustness to the neighborhood size parameter. Evaluations on synthetic and real-world datasets demonstrated high detection accuracy and improved efficiency.

## 3.2 Machine learning techniques

Another approach for detecting Distributed Denial of Service attacks is to use machine learning techniques. These methods involve training a model using past data to recognize and classify future traffic patterns. Unlike statistical techniques that rely on predefined thresholds or distributions, machine learning models can adapt based on the data they are given, making them flexible in detecting evolving types of attacks.

A review by Shyam and Singh [27] outlines the most common machine learning paradigms used in various applications, including network traffic analysis. Similarly, Emmert-Streib and Dehmer [11] provide a taxonomy that divides machine learning into traditional and modern learning paradigms, depending on how data is used during the training process.

The traditional paradigms include:

- **Supervised learning:** This method relies on labeled data, where each traffic sample is marked as either normal or malicious. The model learns to map input features to known labels and can classify new traffic accordingly. It is the most commonly used method in machine learning applications.
- **Unsupervised learning:** Here, the data is not labeled. The model attempts to find patterns or clusters in the data, which may indicate anomalies or unusual behavior. It is useful for discovering unknown attack types.
- **Reinforcement learning:** In this case, an agent learns by interacting with its environment. It receives rewards or penalties based on its actions and gradually learns the best strategy to achieve a goal. Reinforcement learning is less common in DDoS detection but has potential for adaptive defense strategies.

Modern paradigms extend these ideas to handle more complex data scenarios:

- **Semi-Supervised Learning:** Combines a small set of labeled data with a larger set of unlabeled data to improve performance. This is useful when labeling is costly or time-consuming.
- **One-Class Classification:** Trains only on examples of normal traffic. It then flags anything that does not match the normal profile as potentially malicious. This is particularly useful for anomaly detection when attack samples are scarce.
- **Positive-Unlabeled Learning:** In this method, only attack traffic is labeled, while the rest of the traffic is unlabeled. The model learns to distinguish attacks from the rest without needing labeled normal samples.
- **Transfer and Multi-Task Learning:** These approaches allow knowledge from one task or dataset to improve learning on another. This is useful when there is limited training data in one domain, but more data is available elsewhere.
- **Few-Shot and Multi-Label Learning:** Designed for situations where only a small number of labeled examples are available, or where traffic may belong to multiple categories at once.

Some of the machine learning techniques used in recent research are:

- **Random Forest:** Pei et al. [22] developed a method to detect DDoS attacks by first extracting key features from network traffic. They captured attack and normal packets using tools like TcpDump and TFN2K, then identified differences in attributes such as packet sequence numbers and IP patterns. These extracted features were used to train a random forest classifier, which builds an ensemble of decision trees to distinguish between normal and attack traffic. Their experiments demonstrated that the method achieved over 98% detection accuracy with a low false positive rate.
- **AutoEncoder:** Kun Yang et al. [33] propose an unsupervised DDoS detection framework that relies on an AutoEncoder to learn normal traffic patterns. The model is trained solely on normal data and detects anomalies based on reconstruction error. Evaluations on synthetic and public datasets show that it can achieve high detection rates up to 98% with low false positive rates, outperforming traditional anomaly detection methods such as PCA, isolation forest, and one-class SVM. The approach is capable of detecting novel and unknown DDoS attacks without requiring labeled attack data.
- **Logistic Regression and SVM:** Ullah et al. [30] implemented logistic regression and SVM for multiclass DDoS detection using a dataset containing 27 network-related features and five classes, including modern DDoS types such as HTTP flood and SIDDOS. They performed preprocessing by encoding categorical values and applying min-max normalization. Both models achieved a classification accuracy of 98.65%, outperforming prior models on the same dataset. Precision and recall were high across most classes, though both models showed reduced recall for the SIDDOS class due to class imbalance.
- **CNN with Geometrical Metric:** Shieh et al. [26] proposed a DDoS detection framework that combines a convolutional neural network (CNN) with a geometrical metric module to improve detection of unknown attacks. Network flows are represented as  $9 \times 9$  matrices and processed by a CNN to capture spatial patterns. To handle previously unseen attacks, an open-set recognition module calculates density and coverage metrics to identify outliers. An incremental learning mechanism allows the model to be updated with newly labeled samples provided by telecom experts. The model achieved a detection rate over 99% on CICIDS2017 for known attacks, and 99.8% accuracy on CICDDoS2019 after retraining with expert-labeled unknown traffic.
- **Evolutionary KNN with Genetic Algorithm:** Rizvi et al. [24] proposed a hybrid approach that integrates a Genetic Algorithm (GA) for feature selection with several machine learning classifiers, including K-Nearest Neighbors (KNN), to detect DDoS attacks. Using the APA-DDoS dataset, they demonstrated that GA optimization significantly improved classifier performance, with KNN achieving nearly 100% accuracy post-optimization. The GA-enhanced models, particularly KNN, outperformed baseline versions of Logistic Regression and AdaBoost, resulting in accuracy improvements of up to 25%.
- **LSTM:** Liang and Znati [17] propose a deep learning-based DDoS detection framework that leverages a Long Short-Term Memory network to process raw packet header

information, eliminating the need for manual feature engineering. The model learns both temporal and spatial flow-level representations from short sequences of packets, enabling effective detection of DDoS traffic. Evaluations on the CICIDS 2017 dataset show that the LSTM-based scheme outperforms traditional machine learning models trained on statistical features, achieving strong performance even on previously unseen attacks from different days.

### 3.3 LSTM-Autoencoder for anomaly-based DDoS detection

This work uses two key neural network architectures: Long Short-Term Memory networks and autoencoders, which are commonly used for working with sequential data and detecting anomalies without labeled examples.

Recurrent Neural Networks are designed to process sequences by maintaining a hidden state across time steps. However, as Bengio et al. [9] showed, RNNs trained with gradient descent struggle to learn long-term dependencies when the relevant information is far from the point of use. This is known as the vanishing gradient problem and limits their ability to model long-range temporal patterns.

To address this, Hochreiter and Schmidhuber [13] proposed the Long Short-Term Memory architecture. Their design introduced memory cells controlled by input and output gates, enabling the model to retain information over longer periods. Subsequent developments added further gating mechanisms to improve flexibility. Most notably, the forget gate was introduced to allow the model to actively discard outdated information. As noted in the study by Greff et al. [12], the forget gate is now considered essential for stable and efficient training of LSTM networks.

An overview of the LSTM architecture provided by Olah [21] describes the three gates that control information flow. The forget gate decides what information to remove from the cell state, the input gate determines what new information to add, and the output gate selects what information to output. Together, these gates allow LSTMs to keep important information for long periods while discarding what's no longer needed.

Autoencoders are an unsupervised learning technique that use neural networks to learn compressed representations of data. As explained by Bank et al. [8], an autoencoder is a specific type of neural network designed to encode the input into a compressed and meaningful representation, and then decode it back such that the reconstructed input is as similar as possible to the original one. They consist of two main parts, an encoder function that transforms the input into a lower-dimensional representation, and a decoder that attempts to reconstruct the original input from this compressed form. The network is trained by minimizing the reconstruction loss between the input and output, typically using the  $\ell_2$ -norm. A key aspect of autoencoder design is the bottleneck, which restricts the amount of information that can pass through the network and forces it to learn useful structure in the input.

Most notably, in their work, Wei et al. [32] proposed a reconstruction-based LSTM-Autoencoder model for anomaly-based detection of Distributed Denial of Service attacks using multivariate time-series data. Rather than working with raw packet-level traffic, the study focused on flow-based statistical features extracted from the CICDDoS2019 dataset, where traffic flows were preprocessed and stored in CSV files.

The model architecture combines long short-term memory networks with an autoencoder structure. LSTM layers are used in both the encoder and decoder to capture sequential dependencies within network traffic flows. The encoder maps input sequences to a

compressed latent representation, and the decoder reconstructs the original input from this representation. The difference between the original and reconstructed data, measured using Mean Absolute Error (MAE), is used as the anomaly score. A detection threshold is established based on the maximum reconstruction error observed on validation data containing only benign traffic samples.

For feature selection, they used importance scores computed by a Random Forest classifier to select five features: *Max Packet Length*, *Fwd Packet Length Max*, *Fwd Packet Length Min*, *Average Packet Size*, and *Min Packet Length*. These features were used to form multivariate time-series windows of different lengths for training and evaluation.

The LSTM-AE model was trained solely on benign samples, using a split of 70% for training, 10% for validation, and 20% for testing. Training used a batch size of 64, a learning rate of 0.001, and dropout layers set to 0.2 to prevent overfitting. The Adam optimizer was used throughout.

Results showed that the LSTM-AE achieved strong performance across multiple reflection-based attack types. Detection accuracy reached 99.96% for LDAP reflection attacks, 96.08% for DNS reflection attacks, and 96.89% for SNMP reflection attacks when using a 10ms time window. Increasing the window length slightly reduced performance.

Further experiments showed that a batch size of 64 and a learning rate of 0.001 provided the best balance between detection performance and computational efficiency. When compared against classical machine learning techniques such as Decision Trees, Random Forests, and Naive Bayes classifiers applied to the same dataset, the LSTM-AE consistently achieved higher precision, recall, and F1-scores.

Overall, the approach demonstrated that using an LSTM-AE on flow-based multivariate time-series data, without relying on labeled attack traffic during training, is an effective strategy for detecting a range of reflection-based DDoS attacks.

## Chapter 4

# The proposed approach and implementation

This chapter presents the complete architecture and implementation of the proposed DDoS detection system. Section 4.1 provides an overview of the entire system workflow and pipeline, showing how the various components interact during both training and detection phases. Section 4.2 details the preprocessing and feature extraction processes that transform raw network traffic into structured data suitable for deep learning analysis. Section 4.3 explains the approach for creating temporal sequences from variable-length network flows, including both batch and incremental processing capabilities. Section 4.4 describes the LSTM autoencoder neural network architecture, including its dimensionality transformations and training methodology. Finally, section 4.5 explores the threshold determination methods that enable anomaly detection.

### 4.1 System overview and workflow

The core of the detection system is an LSTM-based autoencoder neural network. The thought behind selecting this approach was that if the model is trained exclusively on benign traffic, it would learn to reconstruct normal patterns effectively, but when encountering anomalous traffic during detection, the reconstruction error would be higher, potentially indicating malicious behavior.

The system operates in two distinct phases, training and detection. Figure 4.1 illustrates the complete processing pipeline from raw packet ingestion to final anomaly detection.

During the training phase, benign network traffic in PCAP format is processed through the preprocessing pipeline. The preprocessor divides the traffic into time windows and organizes packets into bidirectional flows based on IP addresses, ports, and protocol information. From each flow, the system extracts 19 features that capture various aspects of network behavior.

These features are arranged into temporal sequences using the adaptive sequence generation techniques described in Section 4.3. Each sequence represents how a flow's characteristics evolve over time. The sequences are normalized using MinMaxScaler, ensuring consistent feature scales.

The LSTM autoencoder then learns to reconstruct these normal traffic patterns. After training, reconstruction errors on validation data are analyzed using multiple methods to

establish detection thresholds. The trained model, thresholds, and scaler parameters are saved for use during detection.

In the detection phase, the system loads the trained model, thresholds, and scaler parameters. New network traffic is processed through the same preprocessing pipeline used during training. Flow features are extracted and arranged into temporal sequences, then normalized using the saved scaler parameters to ensure consistency with the training data.

The autoencoder then generates reconstructions for each sequence. Sequences with errors above the selected threshold are flagged as anomalous. Flows containing any anomalous sequences are marked as an anomaly.

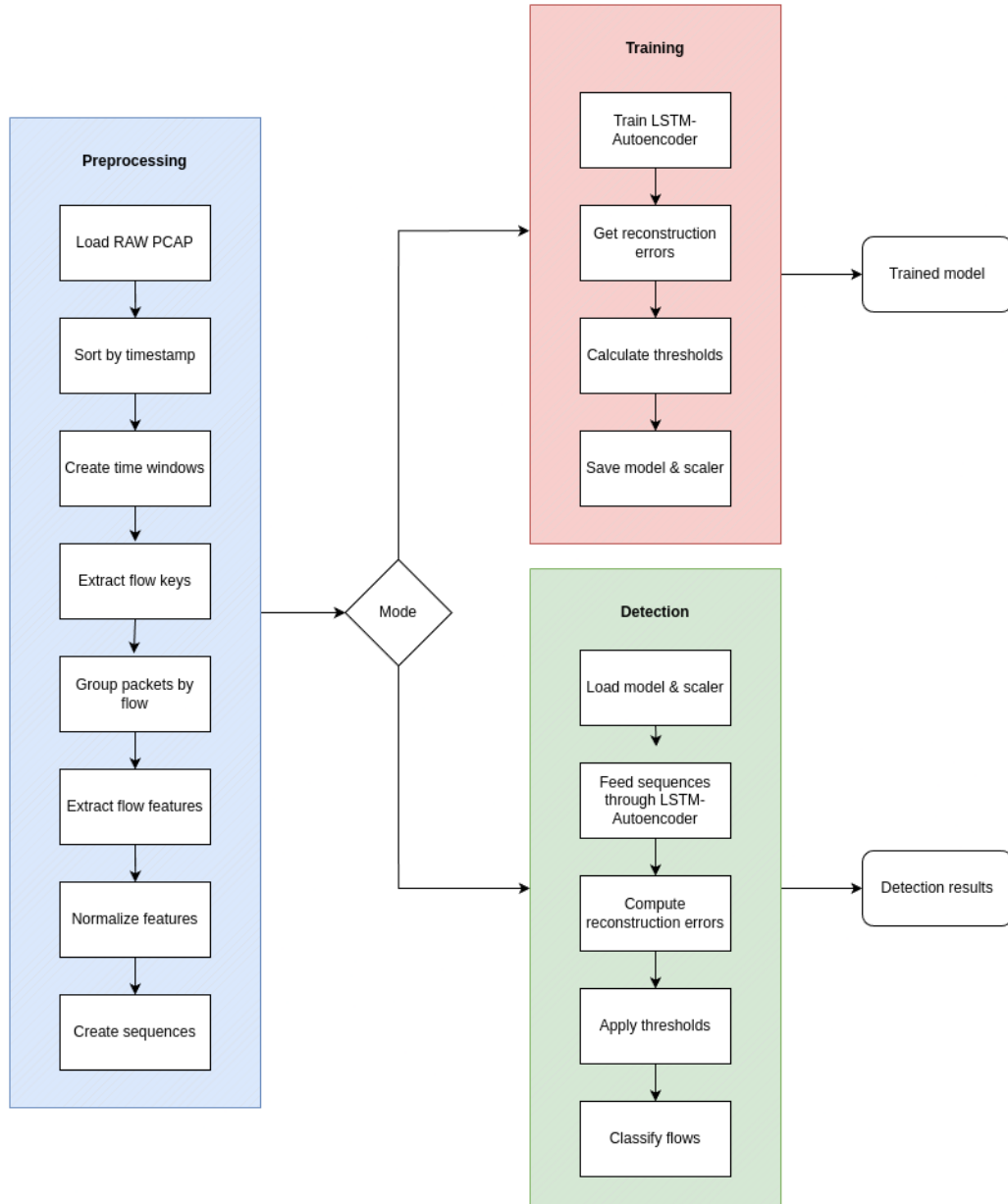


Figure 4.1: Flowchart of the DDoS detection system processing pipeline

## 4.2 Preprocessing and feature extraction

Network traffic analysis starts with PCAP files that contain raw packets. The system uses a preprocessing method to turn this raw data into feature sequences that can be analyzed through deep learning.

For each configurable time window, the system groups packets into bidirectional flows by using a consistent identification scheme that combines source and destination IP addresses, ports, and protocols. This bidirectional approach enables the analysis of network conversations rather than just isolated packets, providing context that might help identify unusual behavior patterns.

When determining the forward and backward directions of traffic, the system maintains a record of the original packet direction seen in each flow. The first time a flow is encountered, the system stores its source-destination relationship in a ‘flow\_directions‘ dictionary. For any subsequent packets in the same flow, this stored direction is used to classify packets as either forward or backward. If the flow has been observed previously in the opposite direction, the system recognizes this and treats it as part of the same bidirectional conversation. Keeping track of direction consistently is essential for protocols where the direction might change during a session.

From each flow within the specified time window, the system extracts 19 network features, which include packet lengths, timing characteristics, and protocol flags. The feature selection was inspired by Sanchez et al. [25], who used ANOVA to identify the most important features for different datasets. The top 10 features from two of those datasets were then combined into a single feature set.

Table 4.1 presents the complete set of features extracted from each network flow.

Feature	Description
Bwd Packet Length Mean	Avg. packet length in backward direction ( $\div 1000$ )
Bwd Segment Size Avg.	Avg. segment size in backward direction ( $\div 1000$ )
Bwd Packet Length Std	Std. dev. of backward packet lengths ( $\div 1000$ )
Bwd Packet Length Max	Max. packet length in backward direction ( $\div 1000$ )
Bwd Packet Length Min	Min. packet length in backward direction ( $\div 1000$ )
Fwd Packet Length Mean	Avg. packet length in forward direction ( $\div 1000$ )
Fwd Packet Length Min	Min. packet length in forward direction ( $\div 1000$ )
Fwd Segment Size Avg.	Avg. segment size in forward direction ( $\div 1000$ )
Fwd IAT Std. Dev.	Std. dev. of inter-arrival times (forward)
Fwd IAT Max	Max. inter-arrival time (forward)
Packet Length Std. Dev.	Std. dev. of packet lengths ( $\div 1000$ )
Packet Length Mean	Avg. packet length (both directions, $\div 1000$ )
Packet Length Min	Min. packet length (both directions, $\div 1000$ )
Max Packet Length	Max. packet length (both directions, $\div 1000$ )
Max Idle Time	Max. time between packets
Mean Idle Time	Avg. time between packets
Download/Upload Ratio	Ratio of bytes (backward/forward)
URG Flag Count	Count of packets with URG flag
CWR Flag Count	Count of packets with CWR flag

Table 4.1: Summary of network flow features used for DDoS detection

All features are normalized with scikit-learn’s `MinMaxScaler` to ensure they contribute equally to the model, regardless of their original scale or measurement units. The scaler’s settings are saved during training and then reused during detection, which helps to keep the normalization consistent throughout the processing stages. This consistency is key to maintaining the effectiveness of the learned model.

### 4.3 Sequence creation and temporal pattern modeling

To capture temporal patterns in network behavior, the system creates sequences that represent how flow characteristics evolve over time. These sequences serve as input to the LSTM autoencoder model, enabling detection of anomalies that manifest over time rather than in isolated packets or windows.

The fundamental challenge I faced was that network flows naturally vary in duration, from brief connections lasting milliseconds to persistent connections spanning minutes, rarely even hours. The thought behind the implementation was to develop an approach that could transform these variable-length flows into fixed-length sequences suitable for deep learning while preserving their temporal characteristics.

To manage this variability while maintaining consistent input dimensions for the neural network, I implemented adaptive sequence creation in the following way:

- **For short flows** spanning fewer than 5 time windows, the system pads the sequence by repeating the last window’s features to reach the required length of 5 windows. For example, if a flow contains only 3 time windows of data, the system duplicates the 3rd window’s features twice to create a complete 5-window sequence. I chose this approach rather than zero-padding because it preserves the most recent traffic characteristics, ensuring the model has meaningful data to work with even for brief connections.
- **For longer flows**, I implemented an adaptive sliding window approach that creates multiple overlapping sequences. The step size between consecutive sequences is calculated using the formula:

$$\text{stride} = \max(1, \min(3, \text{len}(\text{features})//10)) \tag{4.1}$$

This ensures that very short flows use a stride of 1, medium-length flows use a stride proportional to their length, and very long flows never exceed a stride of 3 windows to maintain sufficient overlap.

Figure 4.2 demonstrates how this adaptive approach works in practice for both short and long flows. The upper part shows how short flows are padded by repeating the last window, while the lower part illustrates how long flows are processed with overlapping sequences using an appropriate stride.

```

SHORT FLOW (observed in 3 time windows):
Flow features from windows: [W2] [W7] [W9]
Padded sequence (5):      [W2] [W7] [W9] [W9] [W9]
                           |__|__|
                           repeated

LONG FLOW (observed in 20 time windows):
Flow features from windows: [W1] [W3] [W5] [W8] ... [W53]
Stride = 2

Sequence 1:                [W1] [W3] [W5] [W8] [W10]
Sequence 2:                  [W5] [W8] [W10] [W15] [W17]
Sequence 3:                   [W10] [W15] [W17] [W20] [W23]

And so on...

```

Figure 4.2: Example of sequence creation for short and long flows in batch processing

The reasoning behind this adaptive approach was to balance several competing objectives:

- It ensures all flows contribute to the model regardless of their duration, preventing bias toward flows of any particular length.
- It captures sufficient temporal context from each flow to detect time-dependent anomalies, which is essential for identifying attacks that evolve over time.
- It maintains computational efficiency by limiting the total number of sequences generated from very long flows, preventing memory issues during training and detection.
- It creates overlapping windows that enable detection of anomalies that might appear only during specific phases of a connection’s lifetime.

Many DDoS attacks evolve over time, they might begin with normal-looking connection establishment but then exhibit anomalous patterns during data transfer. By examining multiple overlapping segments of longer flows, the system increases the likelihood of detecting such phase-specific anomalies that might be missed if analyzing the entire flow as a single unit.

The batch processing approach processes PCAPs in chunks rather than loading the entire file at once. In this mode, the system maintains a flow state dictionary that accumulates feature vectors for each flow as they are observed in time windows and processed from the PCAP chunks. After processing all chunks, when all feature vectors for all flows have been collected, the system generates sequences for each flow according to its length using the adaptive approach described earlier. This approach is more memory efficient than loading all raw packets at once but still requires storing all feature vectors in memory until sequence generation is complete.

I also implemented an incremental detection capability using a multi-threaded approach with a detection queue. This implementation is more memory efficient but somewhat slower overall, as it analyzes sequences immediately when they become available rather than waiting to process all data at once.

In the incremental detection scenario, the system still processes network traffic in chunks, but with a key difference in how sequences are handled:

- As soon as a flow accumulates 5 time windows worth of data, the system immediately creates a sequence and queues it for analysis by worker threads.
- For flows that continue beyond 5 windows, the system creates a new sequence each time the flow is observed in additional time windows, while maintaining memory efficiency by only keeping the most recent windows in memory.
- Detection results are accumulated as sequences are processed, with a flow being flagged as anomalous if any of its sequences exceeds the anomaly threshold.

This introduces a distinction between batch and incremental processing. In batch mode, all sequences for a flow are generated only after collecting all its windows, whereas in incremental processing, sequences are created and analyzed as soon as enough windows are available.

Figure 4.3 illustrates how this incremental adaptation works as new windows arrive.

INCREMENTAL PROCESSING:

Flow observed in 5 time windows:

Flow features from windows: [W2] [W5] [W7] [W9] [W10]

Sequence 1: [W2] [W5] [W7] [W9] [W10] -> Queue for analysis

Flow observed in a new window:

Flow features from windows: [W2] [W5] [W7] [W9] [W10] [W12]

Sequence 2: [W5] [W7] [W9] [W10] [W12] -> Queue for analysis

Flow observed in additional windows:

Flow features from windows: [W2] [W5] [W7] [W9] [W10] [W12] [W15] [W18]

Flow features from windows: [W9] [W10] [W12] [W15] [W18] (W2, W5, W7 removed)

Sequence X: [W9] [W10] [W12] [W15] [W18] -> Queue for analysis

Figure 4.3: Example of sequence creation for incremental detection

## 4.4 Model architecture and training

The LSTM autoencoder model analyzes network traffic sequences to learn normal behavior patterns. The bidirectional architecture processes sequences in both forward and backward directions, capturing temporal relationships that might be missed by unidirectional approaches.

The encoder consists of:

- **Input Layer:** Accepts sequences of shape (5, 19) - representing 5 time windows with 19 features each.
- **First Bidirectional LSTM Layer:** Contains 32 units with `return_sequences=True`. This layer processes input sequences in both temporal directions simultaneously.

- **Dropout Layer:** With a rate of 0.2, randomly deactivates 20% of neurons during training to prevent overfitting.
- **Second Bidirectional LSTM Layer:** Contains 16 units (half the size of the first layer), further reducing dimensionality while preserving essential pattern information.
- **Dense Bottleneck Layer:** Contains 8 units with ReLU activation, creating a compressed representation of the input sequence.

The decoder mirrors this structure in reverse:

- **RepeatVector Layer:** Duplicates the bottleneck representation 5 times to match the input sequence length.
- **First LSTM Layer:** Contains 16 units with `return_sequences=True`, beginning the reconstruction process.
- **Dropout Layer:** With a rate of 0.2, maintaining consistent regularization.
- **Second LSTM Layer:** Contains 32 units with `return_sequences=True`, matching the dimensionality of the first encoder layer.
- **TimeDistributed Dense Layer:** Reconstructs the original 19 feature dimensions at each time step.

Figure 4.4 illustrates how data dimensions transform as sequences flow through the network. The input sequence of shape  $(5 \times 19)$  is first expanded to  $(5 \times 64)$  by the bidirectional LSTM layer, which combines forward and backward states. The second bidirectional LSTM collapses the time dimension while producing 32 features. This creates a single vector that captures the entire sequence's temporal context. The dense bottleneck further compresses this to just 8 values representing the most essential patterns in the original sequence. During decoding, the time dimension is restored using RepeatVector, and subsequent LSTM layers gradually reconstruct the feature space back to the original 19 dimensions.

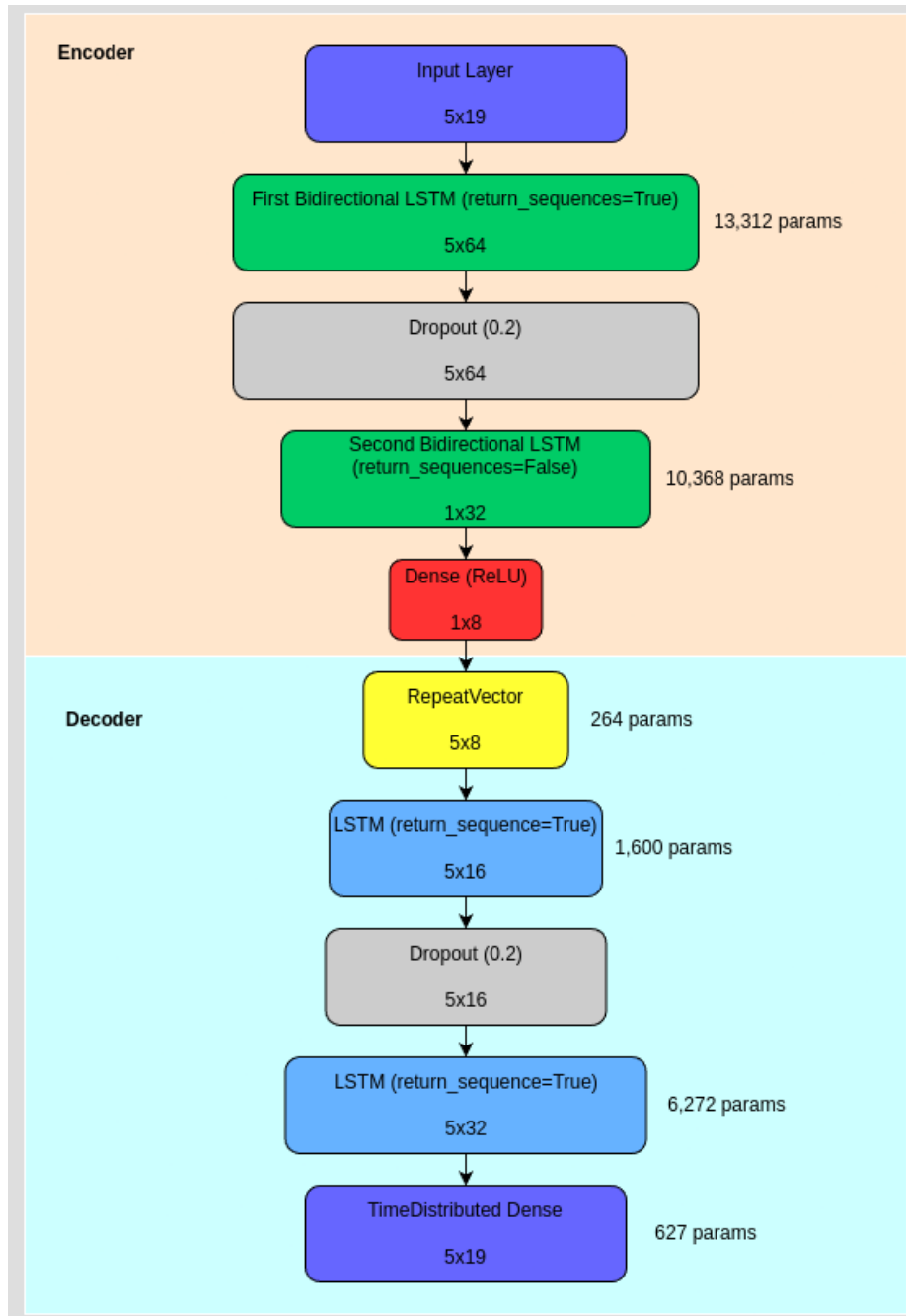


Figure 4.4: Dimension transformation through the LSTM autoencoder

The model contains a total of 32,443 parameters. These parameters represent the weights and biases that the network learns during training to map input sequences to their reconstructions. Training occurs exclusively on benign network traffic, enabling the model to learn normal patterns rather than specific attack signatures. This unsupervised approach may enable the detection of novel attacks not seen during training.

During training, the model uses Adam optimizer with a learning rate of 0.001 and Mean Squared Error (MSE) loss function. MSE was selected because it penalizes larger errors more heavily than smaller ones, making it more sensitive to outliers. The training process

reserves 10% of data for validation to monitor overfitting and continues for up to 100 epochs or until validation loss stops improving.

## 4.5 Detection and threshold determination

For attack detection, the system relies on the reconstruction error between the original and reconstructed flow sequences. When the trained autoencoder attempts to reconstruct a sequence, the resulting error indicates how well that sequence fits the learned normal patterns. Higher reconstruction errors suggest potentially anomalous behavior.

Setting appropriate detection thresholds turned out to be one of the most challenging aspects of the proposed unsupervised anomaly detection. To address this, I implemented three distinct threshold determination methods:

- **Gaussian Mixture Model (GMM) approach:** After training on purely benign traffic, I fit a two-component Gaussian mixture model to the reconstruction errors from the validation set. The model identifies two clusters in the error distribution, one with lower errors representing common traffic patterns, and another with slightly higher errors representing less common but still legitimate patterns. The system then establishes thresholds at specific distances (1.0, 1.5, 2.0, 3.0, and 4.0 standard deviations) from the mean of the lower error component.
- **Percentile-based approach:** This method sets thresholds at specific percentiles of the validation error distribution (90th, 95th, 97.5th, 99th, and 99.5th). This approach directly uses the actual distribution of reconstruction errors observed on normal traffic during validation.
- **Maximum reconstruction-based approach:** This approach calculates thresholds as proportions of the maximum squared reconstruction error observed in the validation data (70%, 80%, 85%, 90%, and 95% of the maximum error).

For each approach, there are five different threshold levels: very low, low, medium, high, and very high. These threshold levels offer a trade-off between detection rate and false positive rate, allowing the system to be tuned according to the specific requirements of the network environment.

The system then identifies anomalies in the following way:

- **Sequence-Level analysis:** Each flow sequence is processed by the autoencoder, and its reconstruction error is calculated using mean squared error. If this error exceeds the selected threshold, the sequence is flagged as anomalous.
- **Flow-Level analysis:** A flow is classified as anomalous if any of its constituent sequences are anomalous. This approach ensures the detection of attacks that might appear only during specific portions of a flow's lifetime.

# Chapter 5

## Experiments and evaluation

This chapter presents the evaluation of the DDoS detection system developed in this thesis. Section 5.1 introduces the datasets used for training and testing, explaining their characteristics and relevance to DDoS detection. Section 5.2 explains the evaluation metrics used to measure detection performance and the approach how the results are evaluated. Section 5.3 analyzes how different threshold levels affect detection performance and explores model reproducibility. Section 5.4 compares the performance of different threshold determination methods. Section 5.5 examines the impact of time window size on detection performance. Section 5.6 investigates how reducing the feature set affects detection capabilities. Section 5.7 evaluates false positive rates on real-world backbone internet traffic. Finally, section 5.8 explores how well the system generalizes across different network environments.

### 5.1 Datasets

Due to the limited availability of high-quality datasets containing real-world DDoS attacks, particularly at the application layer, this evaluation focused on two widely used network traffic datasets, CICIDS2017 [1] and MAWI [2]. These datasets offer contrasting strengths. CICIDS2017 provides labeled attack traffic generated in a controlled lab environment that simulates realistic user behavior and protocol diversity. In contrast, MAWI captures real backbone internet traffic, making it valuable for assessing false positive rates under natural traffic conditions.

The CICIDS2017 dataset was created by the Canadian Institute for Cybersecurity. It includes five days of traffic generated from simulated users interacting over HTTP, HTTPS, FTP, SSH, and email protocols, along with several manually executed attack scenarios. Each day's traffic is provided as a separate PCAP file, and flow-level CSV files were generated using CICFlowMeter.

The following PCAPs were used from CICIDS2017:

- **Training:** Monday PCAP (benign-only traffic)
- **Detection:** Wednesday PCAP (includes multiple DoS attacks and a small number of Heartbleed flows)

The Heartbleed entries were retained due to their very limited number, despite not being traditional DoS attacks.

The class distribution based on ground truth labels from the Wednesday CSV file is shown in Table 5.1.

Flow Type	Count
BENIGN	217823 (95.97%)
DoS Hulk	14108 (6.22%)
DoS GoldenEye	7458 (3.29%)
DoS Slowloris	3873 (1.71%)
DoS Slowhttptest	4216 (1.86%)
Heartbleed	1 (0.00%)
Unique bidirectional flows	226768 (100%)

Table 5.1: Attack distribution in CICIDS2017 Wednesday traffic.

The MAWI dataset was collected from a real internet backbone link as part of the WIDE project. It contains only normal traffic without injected attacks, making it suitable for training and testing anomaly-based detection methods in realistic conditions.

The following PCAPs were used from MAWI:

- **Training:** 202504021400.pcap (April 2, 2025) – 15-minute interval
- **Detection:** 202504031400.pcap (April 3, 2025) – 15-minute interval

The traffic in the MAWI dataset includes common internet protocols such as HTTP, HTTPS, DNS, SMTP, SSH, FTP, and ICMP. The protocol distribution for the training and detection captures is shown in Table 5.2 and Table 5.3.

Protocol	Percentage of Packets (Training)
TCP	57%
UDP	15%
ICMP	15%
IPv6	10%
Others	3%

Table 5.2: Protocol breakdown in MAWI training capture (202504021400.pcap).

Protocol	Percentage of Packets (Detection)
TCP	55%
UDP	15%
ICMP	14%
IPv6	12%
Others	4%

Table 5.3: Protocol breakdown in MAWI detection capture (202504031400.pcap).

In every experiment, during training, 90% of the input data was used for training and the remaining 10% for validation.

The CICDDoS2019 dataset was also considered because of its broad coverage of modern DDoS attack types, including reflection-based and WebDDoS scenarios. However, only a small number of WebDDoS samples were available, and no corresponding flow-level CSV

files were provided for these specific samples, which made them unsuitable for direct evaluation. While the dataset includes various application-layer protocols such as DNS, LDAP, and SNMP—these were primarily used for reflection and protocol exploitation attacks rather than simulating application-layer DDoS attacks.

## 5.2 Evaluation metrics and methodology

To assess the DDoS detection system’s effectiveness, the evaluation uses standard performance metrics that capture different aspects of detection capability. The metrics focus on flow-level detection, as flows represent complete network conversations.

The primary evaluation metrics used in this study are:

- **Accuracy:** The proportion of all network flows correctly classified as either normal or attack. While accuracy provides an overall view of system performance, it can be misleading when classes are imbalanced when normal or attack traffic heavily outnumber the other.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (5.1)$$

- **Detection Rate (Recall):** The proportion of actual attack flows successfully identified by the system. This metric, also called the sensitivity or true positive rate, measures the system’s ability to detect threats. A high detection rate indicates that few attacks go undetected.

$$\text{Detection Rate} = \frac{TP}{TP + FN} \quad (5.2)$$

- **False Positive Rate (FPR):** The proportion of normal flows incorrectly flagged as attacks. This metric is particularly important in operational environments where false alarms can cause unnecessary disruption.

$$\text{FPR} = \frac{FP}{FP + TN} \quad (5.3)$$

- **Precision:** The proportion of flows flagged as attacks that are actually attacks. Precision measures detection reliability, how much we can trust a positive detection result.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5.4)$$

- **F1 Score:** The harmonic mean of precision and recall, providing a single metric that balances detection capability against false alarms. The F1 score is particularly useful when seeking a balanced evaluation of system performance.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5.5)$$

Where:

- *TP* (True Positives): Attack flows correctly identified as attacks
- *TN* (True Negatives): Normal flows correctly identified as normal
- *FP* (False Positives): Normal flows incorrectly identified as attacks
- *FN* (False Negatives): Attack flows incorrectly identified as normal

To evaluate the detection results against ground truth, I needed to account for how network flows are represented in the datasets. The CICIDS2017 CSV files contain multiple entries for the same flow at various timestamps rather than just unique flow identifiers.

I addressed this by collecting bidirectionally unique flows from the CSV files. A canonical flow ID was created for each connection by taking the lexicographically smaller ID between a flow and its reverse direction (swapping source and destination). This ensures that a connection is counted only once regardless of which endpoint initiated it.

For classification purposes, if a flow was labeled as anything other than „BENIGN“ at any point in the ground truth data, the entire flow was counted as an attack flow. The results were then evaluated by determining which canonical flow IDs were flagged as anomalous by the detection system and comparing these against the canonical flow IDs from the ground truth data.

### 5.3 Impact of detection thresholds and model reproducibility

This experiment evaluates how detection performance varies with different GMM-based thresholds and illustrates the reproducibility of results across multiple training and detection runs using the CIC-IDS2017 dataset. A Gaussian Mixture Model was used to determine thresholds at different standard deviation levels from the mean of benign traffic reconstruction errors.

Table 5.4 shows the average detection metrics across four training and detection runs, along with standard deviations to indicate result variability.

Threshold Level	Accuracy	Detection Rate	False Positive Rate	Precision	F1-Score
Very low	89.00% ± 1.01%	99.93% ± 0.01%	12.30% ± 3.04%	36.05% ± 3.08%	52.88% ± 3.10%
Low	93.30% ± 0.38%	99.93% ± 0.01%	7.29% ± 2.14%	47.60% ± 4.06%	64.28% ± 3.92%
Medium	95.94% ± 0.34%	99.92% ± 0.02%	4.61% ± 1.32%	59.35% ± 4.14%	74.16% ± 3.30%
High	98.09% ± 0.13%	99.55% ± 0.42%	2.02% ± 0.47%	76.48% ± 2.36%	86.47% ± 1.20%
Very high	98.84% ± 0.08%	98.33% ± 1.13%	1.18% ± 0.17%	85.05% ± 1.05%	91.13% ± 0.84%

Table 5.4: Average detection metrics with standard deviations across four training and detection runs

The following tables summarize the raw confusion matrix values (TP, FP, TN, FN) for each of the four individual runs. These results highlight how each threshold level behaves across repeated executions.

<b>Threshold Level</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>
Very low	14109	23681	188968	10
Low	14109	14346	198303	10
Medium	14109	9160	203489	10
High	14104	4250	208399	15
Very high	14033	2376	210273	86

Table 5.5: Confusion matrix values for Run 1

<b>Threshold Level</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>
Very low	14109	26071	186578	10
Low	14109	15669	196980	10
Medium	14103	9770	202879	16
High	13953	4370	208279	166
Very high	13502	2370	210279	617

Table 5.6: Confusion matrix values for Run 2

<b>Threshold Level</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>
Very low	14110	37003	175646	9
Low	14109	23258	189391	10
Medium	14108	14407	198242	11
High	14062	5897	206752	57
Very high	13750	2769	209880	369

Table 5.7: Confusion matrix values for Run 3

<b>Threshold Level</b>	<b>TP</b>	<b>FP</b>	<b>TN</b>	<b>FN</b>
Very low	14110	23253	189396	9
Low	14109	13708	198941	10
Medium	14109	8483	204166	10
High	14103	3787	208862	16
Very high	14012	1996	210653	107

Table 5.8: Confusion matrix values for Run 4

These results demonstrate good overall consistency in model performance across multiple training and detection runs. The model shows stable detection rates at lower threshold levels (very low to medium), with minimal variation between runs (standard deviation < 0.02%).

At the very high threshold level, the standard deviation for detection rate increases to 1.13%, showing slightly more variation in how the model handles the boundary between normal and anomalous traffic.

False positive rates display the highest variability (standard deviation = 3.04%) at the very low threshold level, with Run 3 producing notably more false positives under that setting.

The very high threshold level consistently provides the best F1-scores ( $91.13\% \pm 0.84\%$ ) across all runs, indicating this threshold level offers the most reliable overall performance balance.

## 5.4 Comparison of threshold method performance

This experiment shows the observed performance differences between the threshold determination methods described in Section 4.5 when applied to the CIC-IDS2017 Wednesday dataset. All tests were performed using a 100ms time window size, which provided the most granular view of traffic patterns.

Table 5.9 presents the detection results for the three threshold methods across different threshold levels.

Method	Threshold Level	Accuracy	Detection Rate	FPR	Precision	F1-Score	TP	FP
GMM	Very low	89.55%	99.93%	11.14%	37.34%	54.36%	14,109	23,681
	Low	93.67%	99.93%	6.75%	49.58%	66.28%	14,109	14,346
	Medium	95.96%	99.93%	4.31%	60.63%	75.47%	14,109	9,160
	High	98.12%	99.89%	2.00%	76.84%	86.87%	14,104	4,250
	Very high	98.91%	99.39%	1.12%	85.52%	91.94%	14,033	2,376
Simple	Very low	87.98%	99.93%	12.81%	34.12%	50.87%	14,109	27,243
	Low	93.35%	99.93%	7.08%	48.36%	65.18%	14,109	15,064
	Medium	96.34%	99.93%	3.90%	62.98%	77.26%	14,109	8,295
	High	98.28%	99.89%	1.82%	78.43%	87.87%	14,103	3,879
	Very high	98.92%	99.36%	1.11%	85.62%	91.98%	14,028	2,356
Max	Very low	97.49%	60.29%	0.04%	98.92%	74.92%	8,512	93
	Low	96.84%	49.78%	0.03%	99.07%	66.26%	7,028	66
	Medium	96.51%	44.28%	0.03%	99.14%	61.22%	6,252	54
	High	96.19%	39.20%	0.02%	99.18%	56.19%	5,535	46
	Very high	95.90%	34.39%	0.02%	99.24%	51.08%	4,856	37

Table 5.9: Comparison of detection metrics across threshold determination methods (100ms window size).

The results show interesting differences in the behavior of the three threshold methods. The Gaussian Mixture Model GMM and simple percentile-based methods showed nearly identical detection performance, consistently maintaining detection rates above 99% across all threshold levels. However, the maximum reconstruction error-based method showed a fundamentally different pattern, with detection rates decreasing from 60% to 34% as threshold increased.

What proved most challenging was the false positive rate, particularly at lower threshold settings. With the GMM method at very low threshold, I observed an 11.14% false positive rate, equaling to 23,681 false alarms. This high number of false positives would likely overwhelm security teams in operational environments. This suggests that despite the LSTM-autoencoder’s ability to detect attacks, it struggles to completely differentiate some normal traffic patterns from anomalous ones, although it gets significantly better with different thresholds.

The simple percentile-based method performed similarly to GMM but generated even more false positives at lower threshold levels (12.81% FPR at very low threshold).

The maximum reconstruction error-based method’s behavior revealed an interesting tradeoff. It achieved good precision and almost no false positives at all threshold levels, but at the cost of missing many attacks. This demonstrates a critical limitation where many attack flows produce reconstruction errors that, while abnormal, aren’t at the extreme end

of the error distribution. This could be explained by the possibility that some attack traffic shares similar feature characteristics with legitimate traffic, making complete separation using this threshold determination method and the current feature set impossible.

Table 5.10 shows the detection rates for specific DoS attack types at both medium and very high threshold levels for GMM and simple percentile-based methods, and at very low and very high threshold for the max reconstruction error-based method.

Method	Threshold Level	DoS GoldenEye	DoS Hulk	DoS slowloris	DoS Slowhttptest
GMM	Medium	100.0%	100.0%	99.9%	99.8%
	Very high	99.6%	99.5%	99.5%	99.3%
Simple	Medium	100.0%	100.0%	99.9%	99.8%
	Very high	99.6%	99.4%	99.4%	99.2%
Max	Very low	65.2%	60.3%	58.9%	61.3%
	Very high	39.1%	34.4%	31.8%	35.0%

Table 5.10: Detection rates by attack type across different threshold methods (100ms window size)

Looking at the attack-specific detection rates reveals that GMM and simple percentile methods are remarkably consistent across all attack types, with detection rates remaining above 99% even at very high threshold levels.

For the maximum reconstruction error-based method, the detection capability varies significantly by attack type and deteriorates considerably as threshold increases. Slowloris attacks proved most challenging, with detection rates dropping from 58.9% at very low threshold to just 31.8% at very high threshold. This aligns with slowloris’s design as a „low and slow“ attack specifically engineered to evade detection by mimicking normal traffic patterns.

Throughout my experimentation with the 100ms window size and the 19 selected network flow features, the GMM method and the simple percentile-based method, both with very high threshold, offered the best balance between detection capability and false positive rates, with F1 scores of 91.94% and 91.98%, respectively.

## 5.5 Time window size effects

This experiment examines how varying the size of the time window affects the detection performance of DDoS attacks for the CIC-IDS2017 dataset. The time window parameter is fundamental to the detection process, as it determines how network packets are grouped for feature extraction.

Window (s)	Threshold Level	Accuracy	Detection Rate	FPR	Precision	F1-Score	TP	FP	TN	FN
0.1	Very low	89.55%	99.93%	11.14%	37.34%	54.36%	14 109	23 681	188 968	10
	Low	93.67%	99.93%	6.75%	49.58%	66.28%	14 109	14 346	198 303	10
	Medium	95.96%	99.93%	4.31%	60.63%	75.47%	14 109	9 160	203 489	10
	High	98.12%	99.89%	2.00%	76.84%	86.87%	14 104	4 250	208 399	15
	Very high	98.91%	99.39%	1.12%	85.52%	91.94%	14 033	2 376	210 273	86
0.5	Very low	94.96%	99.93%	5.37%	55.29%	71.19%	14 109	11 409	201 240	10
	Low	96.96%	99.93%	3.24%	67.17%	80.34%	14 109	6 895	205 754	10
	Medium	97.97%	99.91%	2.15%	75.49%	86.00%	14 106	4 581	208 068	13
	High	98.95%	99.34%	1.08%	85.99%	92.18%	14 026	2 286	210 363	93
	Very high	99.26%	97.86%	0.64%	90.97%	94.29%	13 817	1 371	211 278	302
1	Very low	96.81%	99.78%	3.39%	66.18%	79.58%	14 088	7 199	205 450	31
	Low	98.04%	99.09%	2.03%	76.40%	86.28%	13 990	4 322	208 327	129
	Medium	98.59%	97.86%	1.36%	82.65%	89.62%	13 817	2 900	209 749	302
	High	99.05%	95.52%	0.71%	89.91%	92.63%	13 486	1 514	211 135	633
	Very high	99.16%	92.90%	0.43%	93.50%	93.20%	13 116	912	211 737	1 003
5	Very low	97.22%	90.71%	2.35%	71.93%	80.24%	12 808	4 998	207 651	1 311
	Low	98.11%	86.12%	1.10%	83.89%	84.99%	12 159	2 335	210 314	1 960
	Medium	98.27%	82.42%	0.68%	89.00%	85.58%	11 637	1 439	211 210	2 482
	High	98.25%	76.03%	0.27%	94.84%	84.40%	10 735	584	212 065	3 384
	Very high	97.99%	70.03%	0.15%	96.83%	81.27%	9 887	324	212 325	4 232

Table 5.11: Detection metrics for various window sizes and threshold levels using GMM (CIC-IDS2017).

Table 5.11 shows clear performance differences across window sizes when using the Gaussian Mixture Model GMM threshold method. The results reveal an interesting trade-off between detection rate and false positives as the window size increases.

With smaller 0.1-second windows at high threshold level, the model achieves 98.12% accuracy while detecting 99.89% of attacks with a 2.00% false positive rate. When increasing to 0.5-second windows at high threshold, the results show improved accuracy 98.95% with a slightly lower but still excellent detection rate of 99.34%, while further reducing false positives to 1.08%.

As the window size increases to 1 second, a performance shift becomes apparent. At medium threshold with 1-second windows, the model achieves 98.59% accuracy with a 97.86% detection rate and only 1.36% false positives. The F1-score 89.62% shows a good balance between precision and recall.

The most dramatic change occurs with 5-second windows, where detection rates drop significantly across all threshold levels. Even at very low threshold, the detection rate falls to 90.71%. At higher threshold levels, the drop becomes more pronounced, with the very high threshold setting detecting 70.03% of attacks.

This pattern shows that temporal granularity plays an important role in DDoS detection. Small windows preserve detailed traffic patterns and anomalies, making attack signatures more distinguishable from normal traffic. The 0.5-second window offers a good balance for this dataset when using GMM thresholds, with the high threshold setting achieving the highest F1-score 92.18% across all configurations.

Larger windows aggregate traffic statistics over longer periods, which can obscure short-lived attack patterns. While they significantly reduce false positives down to just 0.15% with 5-second windows at very high threshold, this comes at the cost of missing more attacks. Many attacks in the CIC-IDS2017 dataset appear to have distinctive patterns that are more easily visible at smaller time scales.

## 5.6 Feature reduction impact on detection performance

This experiment examines how reducing the feature set from 19 to 10 features affects detection performance. The comparison uses a 1-second window size and the GMM threshold method to provide an assessment of how feature selection impacts detection capability.

The 9 features that were excluded from the reduced model were:

- Download/Upload Ratio
- URG Flag Count
- Backward Packet Length Minimum
- CWR Flag Count
- Forward Segment Size Average
- Forward Packet Length Mean
- Forward Packet Length Minimum
- Packet Length Minimum
- Packet Length Mean

Features	Threshold Level	Accuracy	Detection Rate	FPR	Precision	F1-Score	TP	FP	FN
10 Features	Very low	93.02%	98.29%	7.33%	47.08%	63.67%	13,877	15,596	242
	Low	95.45%	96.44%	4.62%	58.11%	72.52%	13,617	9,818	502
	Medium	96.77%	95.13%	3.12%	66.95%	78.59%	13,431	6,631	688
	High	98.13%	93.38%	1.56%	79.95%	86.15%	13,185	3,307	934
	Very high	98.69%	91.78%	0.85%	87.74%	89.71%	12,958	1,811	1,161
19 Features	Very low	96.81%	99.78%	3.39%	66.18%	79.58%	14,088	7,199	31
	Low	98.04%	99.09%	2.03%	76.40%	86.28%	13,990	4,322	129
	Medium	98.59%	97.86%	1.36%	82.65%	89.62%	13,817	2,900	302
	High	99.05%	95.52%	0.71%	89.91%	92.63%	13,486	1,514	633
	Very high	99.16%	92.90%	0.43%	93.50%	93.20%	13,116	912	1,003

Table 5.12: Detection performance comparison between 10-feature and 19-feature models using GMM threshold method (1s window size)

Table 5.12 shows the detection performance differences between the reduced 10-feature model and the full 19-feature model. Both models use the same CICIDS2017 dataset with a 1-second window size and the GMM threshold method.

At all threshold levels, the 19-feature model consistently outperforms the 10-feature model across most metrics. The full feature set achieves better accuracy, lower false positive rates, and higher precision. For instance, at medium threshold, the 19-feature model’s false positive rate 1.36% is less than half that of the reduced model 3.12%.

Detection rates remain high in both models, though the 19-feature model maintains slightly better detection rates at lower threshold settings. At very high threshold, both models show comparable detection rates 91.78% vs 92.90%, suggesting that the most obvious attacks can be detected even with the reduced feature set.

The F1-score, which balances precision and recall, shows a consistent advantage for the 19-feature model. At very high threshold, the 19-feature model achieves an F1-score of 93.20% compared to 89.71% for the 10-feature model.

These results suggest that while the essential 10 CICIDS2017 features capture the core patterns needed for attack detection, the additional 9 features from CICDDoS2019 provide valuable context that helps distinguish borderline cases between normal and attack traffic. This leads to better precision fewer false alarms while maintaining similar detection capability.

## 5.7 Evaluation using the MAWI dataset

The MAWI dataset provides real-world backbone internet traffic that’s ideal for testing how well the system handles legitimate traffic without raising false alarms. This experiment focuses on the model’s ability to recognize normal patterns across different traffic captures from the same network environment.

For this test, the model was trained on one MAWI traffic capture and then evaluated on a completely different MAWI capture from another day. Since MAWI captures contain predominantly benign traffic, this set-up specifically measures the false positive rate, how often normal traffic gets mistakenly flagged as malicious. All results were obtained using the GMM threshold method with a 2-second window size.

Threshold Level	False Positives	True Negatives	Accuracy	False Positive Rate
Very low	23 458	549 862	95.91%	4.09%
Low	19 836	553 484	96.54%	3.46%
Medium	5 593	567 727	99.02%	0.98%
High	2 946	570 374	99.49%	0.51%
Very high	1 933	571 387	99.66%	0.34%

Table 5.13: False positive analysis for model trained on benign MAWI and tested on a different MAWI capture using GMM threshold method and 2s window size

Table 5.13 presents the results of recognizing normal traffic across different captures. At medium threshold level, the false positive rate is 0.98%. The very high threshold level reduces this to 0.34%, showing promising results.

The model distinguishes normal patterns across different traffic captures taken on different days. This suggests it’s learning the fundamental characteristics of legitimate traffic rather than simply memorizing specific examples. This generalization ability is relevant for environments where traffic patterns evolve over time.

The better false positive performance on MAWI compared to CIC-IDS2017 likely stems from the more consistent patterns in the PCAPs.

## 5.8 Cross-Dataset generalization

This experiment explores whether a model trained on one network environment can detect attacks in a completely different environment. For this test, I trained the autoencoder on benign traffic from the MAWI dataset and tested it on the CIC-IDS2017 dataset.

Threshold Level	Accuracy	Detection Rate	FPR	F1-Score	TP	FP	TN	FN
Very low	43.53%	100.00%	99.02%	60.35%	252 672	332 075	3 278	0
Low	43.54%	100.00%	99.00%	60.35%	252 672	332 010	3 343	0
Medium	43.55%	100.00%	98.99%	60.35%	252 672	331 969	3 384	0
High	43.55%	100.00%	98.98%	60.36%	252 672	331 923	3 430	0
Very high	43.56%	100.00%	98.97%	60.36%	252 672	331 903	3 450	0

Table 5.14: Detection metrics of the MAWI-trained model on the CIC-IDS2017 dataset

Table 5.14 shows a clear pattern where the model detected all attacks but also flagged nearly all normal traffic as malicious. This gives poor accuracy around 43.5% despite catching every attack. Different threshold levels made almost no difference.

The main reason for this is simple, the traffic patterns between these environments are too different:

- MAWI contains backbone internet traffic with many diverse services
- CIC-IDS2017 represents network traffic generated in a lab environment with different usage patterns
- The timing and flow characteristics differ significantly between these environments
- Everything from bandwidth to routing architecture creates different traffic patterns

The key takeaway is that for effective anomaly detection, models need to be trained on traffic from the specific environment in which they will be deployed. While the system was able to detect all attacks, it struggled to distinguish between normal traffic from different environments. This confirms that environment-specific training is essential for practical deployment.

## Chapter 6

# Lessons learned and future work

Developing an unsupervised LSTM autoencoder for DDoS detection has highlighted both the potential and the limitations of anomaly-based approaches. One of the most important takeaways is that while unsupervised learning removes the need for labeled attack data and generalizes to unseen threats, it also comes with challenges that must be addressed for real-world deployment.

A key issue is threshold determination. The GMM method offered the best performance in this work, but it relies on the assumption that reconstruction errors follow a Gaussian distribution. This may not hold in every network environment. Furthermore, using fixed multipliers to define threshold levels limits adaptability. A more dynamic thresholding mechanism, perhaps one that updates over time or adapts based on traffic patterns, could help reduce false positives and improve robustness.

The max-error method was shown to be ineffective with mean squared error (MSE), producing very low false positives but missing a large portion of attacks. However, it may become more viable if combined with alternative loss metrics, such as mean absolute error (MAE), which is less sensitive to outliers and could offer better error discrimination.

Feature selection also remains an area for improvement. While this thesis used 19 network flow features, not all contributed effectively to the model's performance. Further research could apply feature selection techniques to identify the most discriminative subset of features, which might lead to improved detection accuracy and fewer false positives with a more lightweight model.

Another limitation is the use of a fixed sequence length of 5 time windows. Although this simplification helped make the model more efficient and easier to train, it may not capture longer-term attack patterns. Future work could explore variable-length sequences or attention-based models that better capture longer temporal dependencies.

The system's ability to generalize was found to be highly dependent on the training environment. When trained and tested within the same dataset, the model performed well. However, it failed to generalize across datasets due to different traffic characteristics. This confirms that any practical deployment of such a system would require training on representative traffic from the target environment.

Adopting this approach into an online detection system would also require further research, but it may not be effective in high-throughput environments due to computational complexity. Future work could explore real-time optimizations or streaming-friendly architectures if live detection is a goal.

Furthermore, the decision rule used in this work, where a flow is marked as anomalous if any one of its sequences is anomalous, may not be optimal. This logic can be overly sensitive

and lead to unnecessary false positives, especially in longer flows. Future improvements could consider alternative strategies like aggregating anomaly scores across sequences to better balance detection sensitivity and precision.

# Chapter 7

## Conclusion

The goal of this thesis was to design, implement, and evaluate an approach for detecting application-layer Denial of Service attacks. To this end, a detection system based on an unsupervised LSTM autoencoder trained on benign traffic was developed.

Chapter 2 provided an overview of DDoS attacks and their development over time. It described the transition from volumetric attacks to techniques targeting application-layer resources and outlined various attack architectures, including centralized and decentralized botnets.

Chapter 3 reviewed existing detection methods, dividing them into statistical and machine learning-based approaches. It introduced the LSTM and autoencoder architectures, which formed the basis of the proposed system, and outlined relevant prior work that motivated the chosen design.

Chapter 4 presented the implementation of the detection system, including the complete data pipeline from PCAP processing to anomaly detection. It explained the feature extraction process, sequence creation and the LSTM autoencoder architecture. Multiple thresholding techniques were introduced to identify anomalous traffic based on reconstruction error.

Chapter 5 evaluated the system on both controlled and real-world datasets. It examined detection performance under different thresholding methods, time window sizes, and feature configurations. The results demonstrated strong detection capabilities within the same environment used for training, while also highlighting limitations in generalization across datasets.

Chapter 6 summarized key observations from development and testing. It outlined challenges related to thresholding, feature relevance, and real-time applicability, and suggested directions for future work, including adaptive thresholds, alternative sequence models, and improved decision logic.

Overall, the method demonstrated promising results in detecting application-layer DDoS attacks without requiring labeled data for training.

# Bibliography

- [1] *Intrusion detection evaluation dataset (CIC-IDS2017)*. Available at: <https://www.unb.ca/cic/datasets/ids-2017.html>.
- [2] *MAWI Working Group Traffic Archive*. Available at: <https://mawi.wide.ad.jp/mawi/>.
- [3] *Five Most Famous DDoS Attacks and Then Some*. 2022. Available at: <https://www.a10networks.com/blog/5-most-famous-ddos-attacks/>.
- [4] *The Evolution of DDoS: Return of the Hacktivists*. 2023. Available at: <https://www.akamai.com/site/en/documents/research-paper/the-evolution-of-ddos-return-of-the-hacktivists.pdf>.
- [5] ADEDEJI, K. B.; ABU MAHFOUZ, A. M. and KURIEN, A. M. DDoS Attack and Detection Methods in Internet-Enabled Networks: Concept, Research Perspectives, and Challenges. *Journal of Sensor and Actuator Networks*, 2023, vol. 12, no. 4, p. 4–5. ISSN 2224-2708. Available at: <https://www.mdpi.com/2224-2708/12/4/51>.
- [6] ALI, B. H.; SULAIMAN, N.; AL HADDAD, S. A. R.; ATAN, R.; HASSAN, S. L. M. et al. Identification of Distributed Denial of Services Anomalies by Using Combination of Entropy and Sequential Probabilities Ratio Test Methods. *Sensors*, 2021, vol. 21, no. 19, p. 1–16. ISSN 1424-8220. Available at: <https://www.mdpi.com/1424-8220/21/19/6453>.
- [7] AMMA, N. G. B.; SELVAKUMAR, S. and VELUSAMY, R. L. A Statistical Approach for Detection of Denial of Service Attacks in Computer Networks. *IEEE Transactions on Network and Service Management*, 2020, vol. 17, no. 4, p. 2511–2522. ISSN 1932-4537. Available at: <https://ieeexplore.ieee.org/document/9188003/>.
- [8] BANK, D.; GIRYES, R. and KOENIGSTEIN, N. Autoencoders. *ArXiv*, 2021, p. 1–17. Available at: <https://arxiv.org/abs/2003.05991>.
- [9] BENGIO, Y.; SIMARD, P. and FRASCONI, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 1994, vol. 5, no. 2, p. 157–166. ISSN 1045-9227. Available at: <https://ieeexplore.ieee.org/document/279181/>.
- [10] BOJOVIĆ, P.; BAŠIČEVIĆ, I.; OCOVAJ, S. and POPOVIĆ, M. A practical approach to detection of distributed denial-of-service attacks using a hybrid detection method. *Computers & Electrical Engineering*, 2019, vol. 73, p. 84–96. ISSN 00457906. Available at: <https://linkinghub.elsevier.com/retrieve/pii/S0045790617327490>.

- [11] EMMERT-STREIB, F. and DEHMER, M. Taxonomy of machine learning paradigms: A data-centric perspective. *WIREs Data Mining and Knowledge Discovery*, 2022, vol. 12, no. 5, p. 1–20. ISSN 1942-4787. Available at: <https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1470>.
- [12] GREFF, K.; SRIVASTAVA, R. K.; KOUTNIK, J.; STEUNEBRINK, B. R. and SCHMIDHUBER, J. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 2017, vol. 28, no. 10, p. 2222–2232. ISSN 2162-237X. Available at: <http://ieeexplore.ieee.org/document/7508408/>.
- [13] HOCHREITER, S. and SCHMIDHUBER, J. Long Short-Term Memory. *Neural Computation*, 1997-11-01, vol. 9, no. 8, p. 1735–1780. ISSN 0899-7667. Available at: <https://direct.mit.edu/neco/article/9/8/1735-1780/6109>.
- [14] HU, W.; GAO, J.; LI, B.; WU, O.; DU, J. et al. Anomaly Detection Using Local Kernel Density Estimation and Context-Based Regression. *IEEE Transactions on Knowledge and Data Engineering*, 2020-2-1, vol. 32, no. 2, p. 218–233. ISSN 1041-4347. Available at: <https://ieeexplore.ieee.org/document/8540843/>.
- [15] HUANG, K.; YANG, L.-X.; YANG, X.; XIANG, Y. and TANG, Y. Y. A Low-Cost Distributed Denial-of-Service Attack Architecture. *IEEE Access*, 2020, vol. 8, p. 42111–42119. ISSN 2169-3536. Available at: <https://ieeexplore.ieee.org/document/9018054/>.
- [16] KHALAF, B. A.; MOSTAFA, S. A.; MUSTAPHA, A.; MOHAMMED, M. A. and ABDUALLAH, W. M. Comprehensive Review of Artificial Intelligence and Statistical Approaches in Distributed Denial of Service Attack and Defense Methods. *IEEE Access*, 2019, vol. 7, p. 51691–51713. ISSN 2169-3536. Available at: <https://ieeexplore.ieee.org/document/8692706/>.
- [17] LIANG, X. and ZNATI, T. A Long Short-Term Memory Enabled Framework for DDoS Detection. In: *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, p. 1–6. ISBN 978-1-7281-0962-6. Available at: <https://ieeexplore.ieee.org/document/9013450/>.
- [18] MAJED, H.; NOURA, H.; SALMAN, O.; MALLI, M. and CHEHAB, A. Efficient and Secure Statistical DDoS Detection Scheme. In: *Proceedings of the 17th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science and Technology Publications, 2020-7-8, p. 153–161. ISBN 978-989-758-445-9. Available at: <https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0009873801530161>.
- [19] MANTAS, G.; STAKHANOVA, N.; GONZALEZ, H.; JAZI, H. H. and GHORBANI, A. A. Application-layer denial of service attacks: taxonomy and survey. *International Journal of Information and Computer Security*, 2015, vol. 7, 2/3/4, p. 1–17. ISSN 1744-1765. Available at: <http://www.inderscience.com/link.php?id=73028>.
- [20] MERKEBAIULY, M. Overview of Distributed Denial of Service (DDoS) attack types and mitigation methods. *InterConf*, 2024-03-19, 43(193), p. 494–508. ISSN 2709-4685. Available at: <https://archive.interconf.center/index.php/2709-4685/article/view/5690>.

- [21] OLAH, C. *Understanding LSTM Networks*. 2015. Available at: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [22] PEI, J.; CHEN, Y. and JI, W. A DDoS Attack Detection Method Based on Machine Learning. *Journal of Physics: Conference Series*, 2019-06-01, vol. 1237, no. 3, p. 1–5. ISSN 1742-6588. Available at: <https://iopscience.iop.org/article/10.1088/1742-6596/1237/3/032040>.
- [23] PRASEED, A. and THILAGAM, P. S. DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications. *IEEE Communications Surveys & Tutorials*, 2019, vol. 21, no. 1, p. 661–685. ISSN 1553-877X. Available at: <https://ieeexplore.ieee.org/document/8466561/>.
- [24] RIZVI, F.; SHARMA, R.; SHARMA, N.; RAKHRA, M.; ALEDAILY, A. N. et al. An evolutionary KNN model for DDoS assault detection using genetic algorithm based optimization. *Multimedia Tools and Applications*, 2024, vol. 83, no. 35, p. 83005–83028. ISSN 1573-7721. Available at: <https://link.springer.com/10.1007/s11042-024-18744-5>.
- [25] SANCHEZ, O. R.; REPETTO, M.; CARREGA, A.; BOLLA, R. and PAJO, J. F. Feature Selection Evaluation towards a Lightweight Deep Learning DDoS Detector. In: *ICC 2021 - IEEE International Conference on Communications*. IEEE, 2021, p. 1–6. ISBN 978-1-7281-7122-7. Available at: <https://ieeexplore.ieee.org/document/9500458/>.
- [26] SHIEH, C.-S.; NGUYEN, T.-T. and HORNG, M.-F. Detection of Unknown DDoS Attack Using Convolutional Neural Networks Featuring Geometrical Metric. *Mathematics*, 2023, vol. 11, no. 9, p. 1–22. ISSN 2227-7390. Available at: <https://www.mdpi.com/2227-7390/11/9/2145>.
- [27] SHYAM, R. and SINGH, R. *Journal of Advancements in Robotics: A Taxonomy of Machine Learning Techniques*. 2021. ISSN 24551872. Available at: [https://www.researchgate.net/publication/358089496\\_A\\_Taxonomy\\_of\\_Machine\\_Learning\\_Techniques](https://www.researchgate.net/publication/358089496_A_Taxonomy_of_Machine_Learning_Techniques).
- [28] TAN, Z.; JAMDAGNI, A.; HE, X.; NANDA, P. and LIU, R. P. A System for Denial-of-Service Attack Detection Based on Multivariate Correlation Analysis. *IEEE Transactions on Parallel and Distributed Systems*, 2014, vol. 25, no. 2, p. 447–456. ISSN 1045-9219. Available at: <http://ieeexplore.ieee.org/document/6519239/>.
- [29] TRIPATHI, N. and HUBBALLI, N. Application Layer Denial-of-Service Attacks and Defense Mechanisms. *ACM Computing Surveys*, 2022-05-31, vol. 54, no. 4, p. 1–33. ISSN 0360-0300. Available at: <https://dl.acm.org/doi/10.1145/3448291>.
- [30] ULLAH, M. A.; JAMAL, A. A.; TUHIN, R. A. and AKHTER, S. DETECTING DISTRIBUTED DENIAL OF SERVICE ATTACKS USING LOGISTIC REGRESSION AND SVM METHODS. *ArXiv*, 2024, p. 1–6. Available at: <https://arxiv.org/abs/2411.14512>.
- [31] WANG, J.; YU, L.; LUI, J. C. and LUO, X. Modern DDoS Threats and Countermeasures: Insights into Emerging Attacks and Detection Strategies. *ArXiv*, 2025, p. 1–15. Available at: <https://arxiv.org/abs/2502.19996>.

- [32] WEI, Y.; JANG JACCARD, J.; SABRINA, F.; XU, W.; CAMTEPE, S. et al. Reconstruction-based LSTM-Autoencoder for Anomaly-based DDoS Attack Detection over Multivariate Time-Series Data. *ArXiv*, 2021, vol. 14, no. 8, p. 1–13. Available at: <https://arxiv.org/abs/2305.09475>.
- [33] YANG, K.; ZHANG, J.; XU, Y. and CHAO, J. DDoS Attacks Detection with AutoEncoder. In: *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2020, p. 1–9. ISBN 978-1-7281-4973-8. Available at: <https://ieeexplore.ieee.org/document/9110372/>.