Master of Science in Telecommunication Systems
May 2025

HTTP Flood Attack Detection using Machine Learning, Deep Learning and Explainable AI

Prajna Bala Sai Potnuri

Sri Phani Deverakonda

Faculty of Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden

This thesis is submitted to the Faculty of Engineering at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Telecommunication Systems. The thesis is equivalent to 20 weeks of full-time studies.

The authors declare that they are the sole authors of this thesis and that they have not used any sources other than those listed in the bibliography and identified as references. They further declare that they have not submitted this thesis at any other institution to obtain a degree.

Contact Information:
Author(s):
Prajna Bala Sai Potnuri
E-mail: prpo21@student.bth.se

Sri Phani Deverakonda
E-mail: srdv21@student.bth.se

University advisor:
Dragos Ilie
E-mail: dragos.ilie@bth.se
Department of Computer Science

Faculty of Engineering
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet	:	www.bth.se
Phone	:	+46 455 38 50 00
Fax	:	+46 455 38 50 57

Abstract

Background: Distributed Denial-of-Service (DDoS) attacks, particularly HTTP Flood Attacks, pose a severe threat to the availability and performance of web-based services. These attacks exploit HTTP methods to flood servers, often producing traffic patterns that differ from normal user behavior. As attack patterns evolve, there is an increasing need for intelligent and adaptable detection techniques that not only identify threats accurately but also provide explainable insights into their decision-making processes.

Objectives: This thesis investigates the effectiveness of machine learning and deep learning models for detecting HTTP Flood Attacks. It also explores how explainable AI techniques can improve model interpretability and guide the selection of the most suitable models for real-time deployment.

Methods: The study evaluates seven supervised models: Logistic Regression, Support Vector Machine, Random Forest, k-Nearest Neighbors, LightGBM, Convolutional Neural Network, and Long Short-Term Memory. The CICIDS 2017 dataset was used to train and test the models. Performance was assessed using accuracy, precision, recall, and F1-score. SHAP (SHapley Additive Explanations) was applied to analyze the importance of input features and understand the models' prediction behavior.

Results: The k-Nearest Neighbors (KNN) model achieved the best overall performance with an F1-score of 0.977 and balanced precision and recall, demonstrating strong generalization and minimal misclassifications. LightGBM also performed competitively with high accuracy and computational efficiency. In contrast, deep learning models such as CNN and LSTM exhibited higher false positive rates due to over-reliance on a single feature. SHAP analysis highlighted that models with balanced feature importance distributions, such as KNN and LightGBM, were more interpretable and reliable.

Conclusions: Machine learning models, particularly KNN and LightGBM, offer effective solutions for detecting HTTP Flood Attacks, outperforming deep learning models in accuracy, explainability, and real-time feasibility. Explainable AI plays a vital role in building trust, reducing false positives, and enhancing the transparency of detection systems. Future work should focus on live deployment, adaptive learning mechanisms, and the development of automated response strategies.

Keywords: HTTP Flood Attacks, Machine Learning, Explainable AI, SHAP, Intrusion Detection

Acknowledgments

First and foremost, we would like to express our sincere gratitude to our thesis advisor, Dr. Dragos Ilie, for his continuous support, guidance, and encouragement throughout the course of this research. His expertise, insightful feedback, and thoughtful suggestions were invaluable and played a key role in the successful completion of this thesis.

We would also like to extend our appreciation to the faculty and staff of the Department of Computer Science at Blekinge Institute of Technology for providing a strong academic foundation, access to essential resources, and an environment that fostered research and innovation.

Our heartfelt thanks go to our friends and fellow students for their constant support, motivation, and for being part of this journey. Their feedback, discussions, and camaraderie made the research process more collaborative and enriching.

Lastly, we are deeply thankful to our families for their endless love, patience, and encouragement. Their belief in us has been our greatest source of strength throughout our academic journey.

- Prajna Bala Sai Potnuri
- Sri Phani Deverakonda

Contents

List of Figures

List of Tables

List of Abbreviations

ACK	Acknowledgment (TCP flag)
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under the Curve
Bwd	Backward (refers to traffic direction in packet features)
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DL	Deep Learning
DNN	Deep Neural Network
DNS	Domain Name System
DoS	Denial of Service
DT	Decision Tree
DDoS	Distributed Denial of Service
FN	False Negative
FP	False Positive
GET	HTTP GET (request method)
HTTP	HyperText Transfer Protocol
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IDPS	Intrusion Detection and Prevention System
IoT	Internet of Things

KNN	K-Nearest Neighbors
LGB	Light Gradient Boosting Machine (LightGBM)
LIME	Local Interpretable Model-agnostic Explanations
LR	Logistic Regression
LSTM	Long Short-Term Memory
MCCNN	Multi-Channel Convolutional Neural Network
ML	Machine Learning
NN	Neural Network
NTP	Network Time Protocol
POST	HTTP POST (request method)
RAM	Random Access Memory
RF	Random Forest
RFC	Random Forest Classifier
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
AUC	Receiver Operating Characteristic
SDN	Software-Defined Networking
SHAP	SHapley Additive Explanations
SVM	Support Vector Machine
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
TinyML	Tiny Machine Learning
UDP	User Datagram Protocol
U2R	User to Root (attack category)
XAI	Explainable Artificial Intelligence

Chapter 1

Introduction

In the era of pervasive connectivity and digital dependency, the threat landscape for cyberattacks has evolved dramatically. Distributed Denial of Service (DDoS) attacks have emerged as a formidable adversary in the cybersecurity domain, posing significant challenges to organizations worldwide [66] [68]. A DDoS attack is an attempt to disrupt the normal functioning of a targeted server, service, or network by overwhelming it with a flood of Internet traffic. These attacks leverage the power of multiple compromised devices, often organized into a botnet, to generate massive volumes of malicious traffic that paralyze targeted systems.

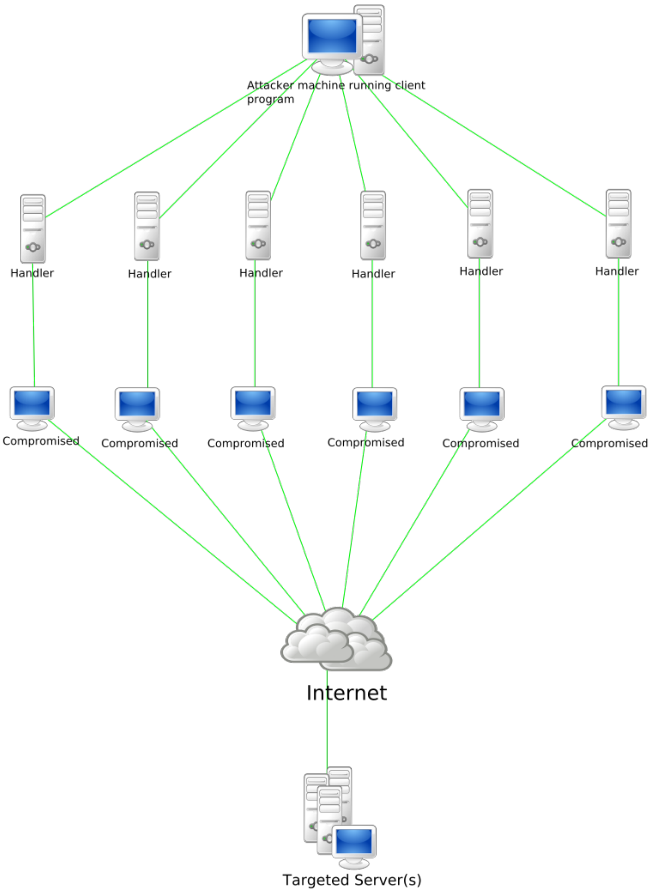

Figure 1.1: Botnet-based DDoS Attack Architecture [3].

From the figure 1.1, we observe a hierarchical structure in which an attacker remotely controls several intermediate systems (known as handlers or masters), which in turn manage a network of infected bots (also referred to as zombies). These bots coordinate to flood the victim's network with traffic, ultimately exhausting its resources and causing service disruption. This architecture exemplifies how attackers scale their operations and evade direct detection by using a multi-tier command-and-control structure, where intermediate handlers manage large groups of bots on the attacker's behalf [3].

Network security is essential because of our reliance on the internet for almost everything, from communication to shopping. Protecting digital systems is critical to keep them running smoothly and safely. Cyberattacks, particularly DDoS attacks, are a growing concern as they become more sophisticated and prevalent [4]. These attacks not only disrupt services but also result in the theft of sensitive information and significant financial losses. Consequently, improving network security measures is a top priority to safeguard digital infrastructure.

The scale of DDoS attacks has surged dramatically over the last decade. Reports indicate a staggering 807% increase in DDoS attacks between 2013 and 2022, culminating in 13 million attacks globally in 2022 alone. The financial repercussions are equally alarming, with small businesses incurring an average loss of $52,000 per incident and enterprises suffering losses of approximately $444,000. High-profile targets, such as the United States, account for 18.3% of global attacks, highlighting the widespread nature of this cyber threat. This exponential rise in attack frequency underscores the urgent need for robust mitigation measures. As illustrated in Figure 1.2, this significant growth reflects the evolving landscape of DDoS threats, emphasizing the importance of proactive security strategies [2].

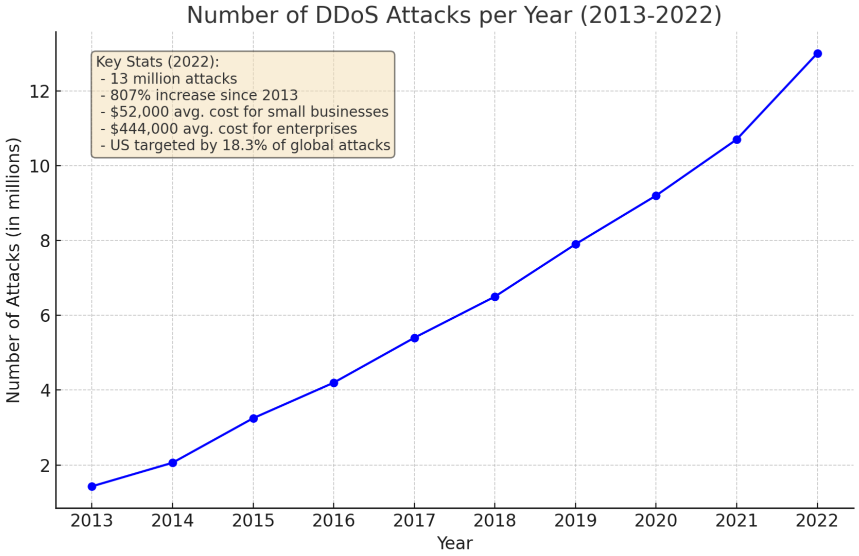

Figure 1.2: Annual Number of DDoS Attacks from 2013 to 2022 [2]

Adding to this, the market for DDoS protection reflects the growing awareness and investment in this area, projected to expand from \$2.91 billion in 2022 to \$7.45 billion by 2030. Such growth underscores the pressing demand for innovative and scalable solutions to counter these threats. Recent studies indicate a notable increase in HTTP flood attacks, which have become a prominent subset of DDoS attacks. HTTP flood attacks are particularly concerning due to their ability to mimic legitimate traffic, making them difficult to detect and mitigate. As highlighted in recent statistics, these attacks are responsible for a significant portion of modern DDoS activity, underscoring the importance of focused research and countermeasures to address this growing threat effectively [2].

1.1 Motivation Behind the Research

The reason for this research comes from the increasing frequency and complexity of HTTP flood attacks, a specific type of DDoS attack that creates serious challenges for modern cybersecurity. With these attacks becoming more common and capable of mimicking genuine traffic, current detection methods often struggle to identify and prevent them effectively.

HTTP flood attacks specifically target important industries like finance, healthcare, and e-commerce, where continuous service availability is crucial [60]. As society increasingly relies on digital systems for everyday activities, even a short disruption caused by such an attack can lead to significant financial losses, data breaches, and harm to a company's reputation. For example, recent cases show that businesses can lose millions of dollars due to long outages caused by undetected or poorly managed HTTP flood attacks [5].

Current methods to prevent these attacks often rely on traditional systems that use fixed rules or look for unusual behavior. However, these approaches often fail to keep up with the ever-changing tactics used by attackers [73]. This project plans to solve this problem by using machine learning and deep learning methods, along with Explainable AI (XAI), to build a strong and transparent detection system. XAI ensures that the system's decisions are understandable, helping cybersecurity teams trust and improve the models [6].

Additionally, this project is driven by the need to create detection systems that can grow and adapt. As cyber threats continue to change, systems that stay the same over time are not effective. By training machine learning models on varied datasets, this project aims to create a flexible solution that can find complex patterns that show an HTTP flood attack is happening. Comparing different models will also help choose the best and most efficient one for real-world use.

The main goal of this project is to contribute to cybersecurity by providing a practical, explainable, and flexible way to combat HTTP flood attacks. This research not only focuses on improving detection success but also hopes to encourage future advancements in DDoS prevention methods.

1.2 Research Gap

Although a significant amount of research has been conducted on detecting Distributed Denial of Service (DDoS) attacks using machine learning techniques, the majority of these efforts have focused on traditional or volumetric attack types at the network layer. In contrast, application-layer attacks such as HTTP Flood Attacks remain comparatively underexplored. Existing approaches often prioritize achieving high accuracy but lack attention to interpretability and explainability, which are essential for practical cybersecurity applications.

Furthermore, while various machine learning and deep learning models have been evaluated in previous work, many studies do not address the real-world need for transparent decision-making. The absence of Explainable AI (XAI) techniques, particularly SHAP-based analysis, limits the operational trust and adoption of such models. This study aims to bridge this gap by investigating interpretable machine learning models tailored specifically for HTTP flood detection and assessing their performance and transparency using SHAP-based feature analysis.

1.3 Aim and Objectives

The purpose of this study is to investigate HTTP flood attacks, a specific type of Distributed Denial of Service (DDoS) attack, and evaluate the effectiveness of machine learning techniques in detecting such attacks. By incorporating Explainable AI (XAI) methodologies, this thesis aims to enhance the interpretability of detection models and provide transparency into their decision-making processes. The objective is to identify the most effective machine learning models and demonstrate their practical application in combating HTTP flood attacks. In this work, HTTP flood detection is approached as a binary classification problem, where network traffic is categorized as either normal or malicious.

To accomplish the main goal of this research, the following specific objectives are formulated:

1. **To investigate how to detect HTTP flood attacks using machine learning and deep learning techniques.**
   Explore and evaluate existing machine learning algorithms to identify HTTP flood attacks. Adapt and refine these algorithms to enhance their accuracy and scalability in detecting complex patterns indicative of such attacks in real-world scenarios.

2. **To compare the effectiveness of machine learning and deep learning models for HTTP flood attack detection.**
   Conduct a comparative analysis of various machine learning and deep learning models to identify their strengths and limitations. This includes assessing performance metrics such as precision, recall, and accuracy to determine the most suitable model for detecting HTTP flood attacks.

3. **To integrate Explainable AI (XAI) for transparent and interpretable HTTP flood detection models.**

Utilize SHAP (Shapley Additive Explanations) and other XAI tools to interpret and explain the decision-making process of the proposed models. This ensures the reliability, trustworthiness, and ethical use of AI-based solutions in cybersecurity.

4. **To provide a discussion on the insights derived from model comparisons and explainability results.**
   Analyze the outcomes of the detection models and their explanations to derive meaningful insights. Discuss the implications of the results for future research and practical applications in the field of network security.

1.4 Research Questions

The following research questions are formulated to achieve the above objectives:

R1 **To what extent can machine learning and deep learning models effectively detect HTTP flood attacks?**
To assess the capabilities of various machine learning and deep learning models in identifying HTTP flood attacks and evaluate their performance based on metrics such as precision, recall, and accuracy.

R2 **How can Explainable AI (XAI) enhance the interpretability of HTTP flood attack detection models?**
To utilize XAI techniques, such as SHAP, to interpret the decision-making process of machine learning and deep learning models, ensuring transparency and reliability in the detection process.

R3 **What are the key insights derived from the comparative analysis of detection models in combating HTTP flood attacks?**
To analyze the comparative performance of machine learning and deep learning models and derive actionable insights that can guide the development of effective detection strategies for HTTP flood attacks.

1.5 Outline of the Thesis

The structure of this thesis is organized into sequential chapters, each focusing on distinct aspects of the research as detailed below:

1. **Chapter 2: Background Technologies**
   This chapter provides an introduction to the fundamental technologies related to the research, including an overview of DDoS attacks, HTTP flood attacks, and the integration of machine learning and deep learning techniques. The focus will be on their relevance and application within the context of this research.

2. **Chapter 3: Related Work**
   This chapter presents a critical analysis of existing research concerning DDoS

attacks, HTTP flood detection mechanisms, and the application of Explainable AI (XAI) in cybersecurity. The chapter aims to highlight the gaps in the current literature and establish the foundation for the research methodology.

3. **Chapter 4: Methodology**
   A comprehensive description of the research design is presented, encompassing the dataset preparation, selection of machine learning and deep learning models, implementation details, and evaluation metrics. This chapter outlines the experimental setup and the tools utilized for data analysis and interpretation.

4. **Chapter 5: Results and Analysis**
   This chapter discusses the results derived from the implemented methodologies. It includes the comparative performance of machine learning and deep learning models, along with visual representations of data through graphs and tables.

5. **Chapter 6: Discussion**
   The findings from the experiments are discussed in depth, with a systematic report on the results and a detailed interpretation of their implications. This chapter connects the research outcomes with the theoretical framework established in earlier chapters.

6. **Chapter 7: Conclusion and Future Work**
   The thesis concludes by summarizing the research findings and reflecting on their implications for the field. Recommendations for future research directions are also provided, suggesting areas where further investigation could be beneficial.

Chapter 2

Background Technologies

2.1 Distributed Denial of Service (DDoS) Attacks

Distributed Denial of Service (DDoS) attacks represent one of the most pervasive and disruptive threats in the realm of cybersecurity. These attacks aim to overwhelm a target server, service, or network by inundating it with an immense volume of traffic, rendering it inaccessible to legitimate users. Unlike traditional Denial of Service (DoS) attacks, which rely on a single source to flood the target, DDoS attacks employ a distributed network of compromised devices, often referred to as botnets. This distributed nature amplifies the scale of the attack, making it more difficult to mitigate and trace back to the attacker [23].

2.1.1 Mechanism of DDoS Attacks

The fundamental mechanism of a DDoS attack lies in exploiting the resource limitations of the target system. By sending more requests or data packets than the target can handle, the attacker effectively exhausts its bandwidth, computational power, or memory. The attack process typically involves the following phases:

- **Target Identification**: The attacker selects a target, which could be a website, server, or network. Often, high-value targets include e-commerce platforms, financial institutions, and government websites.

- **Botnet Recruitment**: To execute a large-scale attack, the attacker compromises numerous devices, including computers, IoT devices, and smartphones, by deploying malware. These infected devices form a botnet, which can be remotely controlled.

- **Command and Control (C&C)**: The attacker uses a central control mechanism to issue commands to the botnet devices, instructing them to begin the attack.

- **Attack Execution**: The botnet devices send an overwhelming volume of traffic or requests to the target, exploiting vulnerabilities and consuming resources.

- **Impact**: The target becomes overwhelmed, resulting in slowed performance or complete inaccessibility for legitimate users.

The orchestration of these phases highlights a high degree of coordination and technical sophistication. Attackers often leverage anonymization techniques such as encrypted communication, IP spoofing, and proxy routing to obfuscate their identity and avoid attribution.

This multistage progression aligns with well-established cybersecurity frameworks, such as the Cyber Kill Chain [34], MITRE ATT&CK [61], [62], and Diamond Model [15]. For example, the "Target Identification" and "Botnet Recruitment" stages resemble the Reconnaissance and Weaponization phases in the Cyber Kill Chain. Mapping DDoS attack mechanisms to these models enhances our understanding of adversarial behavior and informs the development of robust detection and defense strategies.

2.1.2 Types of DDoS Attacks

DDoS attacks are categorized based on the techniques and goals used by the attackers. The three primary types are volumetric attacks, protocol attacks, and application-layer attacks.

Volumetric Attacks

Volumetric attacks aim to overwhelm the target's network bandwidth by sending a massive volume of data packets. These attacks are often executed using reflection and amplification techniques to magnify the attack traffic [28].

- **UDP Floods**: Attackers send large amounts of UDP packets to random ports on the target, causing the server to repeatedly check for nonexistent applications.

- **DNS Amplification**: Exploiting misconfigured DNS servers, attackers send small queries that generate significantly larger responses, which are directed at the target.

- **NTP Amplification**: Similar to DNS amplification, attackers use Network Time Protocol (NTP) servers to amplify traffic directed at the target.

Protocol Attacks

Protocol attacks exploit vulnerabilities in the communication protocols used by the target. These attacks consume server resources such as connection tables or session buffers [69]

- **SYN Floods**: Exploiting the TCP handshake process, attackers send SYN packets but never complete the handshake by sending ACK packets, leaving the server with half-open connections.

- **Ping of Death**: Oversized or malformed ICMP packets are sent to the target, causing crashes or instability.

- **Smurf Attacks**: Attackers spoof the target's IP address and send ICMP requests to a network, causing multiple devices to reply to the target, overwhelming it with responses.

Application-Layer Attacks

Application-layer attacks target specific applications or services. These attacks generate traffic that mimics legitimate user behavior, making them difficult to detect and filter [26].

- **HTTP Floods**: Attackers send excessive HTTP GET or POST requests, consuming server resources and rendering the service inaccessible.
- **Slowloris**: The attacker opens numerous HTTP connections to the server but sends data slowly, keeping the connections open for an extended period and exhausting server resources.
- **DNS Query Floods**: Targeting DNS servers, attackers flood them with queries, preventing legitimate DNS requests from being resolved.

2.1.3 Impact of DDoS Attacks

The impact of DDoS attacks is extensive, affecting businesses, governments, and individuals alike. The consequences of these attacks can be broadly categorized into the following areas [11]:

- **Financial Losses**: Prolonged downtime can result in significant revenue losses for businesses, particularly in sectors like e-commerce and finance. Studies have shown that large-scale DDoS attacks can cost enterprises millions of dollars in lost sales and recovery efforts.
- **Reputational Damage**: Organizations that experience frequent or prolonged outages due to DDoS attacks risk losing customer trust. This can be particularly damaging for businesses that rely on their online presence to maintain their brand image.
- **Operational Disruptions**: DDoS attacks can cripple critical infrastructure, such as healthcare systems, transportation networks, and financial services, disrupting day-to-day operations and potentially endangering lives.
- **Resource Allocation**: Organizations must allocate significant resources to mitigate DDoS attacks, including deploying advanced security solutions and maintaining standby servers to handle increased traffic.

2.1.4 HTTP Flood Attacks

HTTP flood attacks are a subset of application-layer attacks that target the web server by sending a deluge of HTTP requests. Unlike volumetric attacks that focus on overwhelming the network bandwidth, HTTP floods consume server resources such as CPU and memory, leading to performance degradation or complete service unavailability.

Mechanism

Attackers use botnets to generate an overwhelming number of HTTP GET or POST requests, targeting specific endpoints or randomly selecting resources. By simulating legitimate user behavior, these attacks bypass traditional rate-limiting and filtering mechanisms.

Motivations

HTTP flood attacks are a form of application-layer DDoS attack in which the attacker attempts to exhaust a web server's resources by sending a high volume of seemingly legitimate HTTP requests. Unlike traditional volumetric DDoS attacks that target bandwidth or lower-layer services, HTTP floods focus on overwhelming the application itself—such as login portals, search forms, APIs, or dynamic content endpoints. Attackers may be motivated by economic disruption (e.g., making a service unavailable during peak business hours), competitive sabotage, or ideological objectives. Since HTTP floods mimic real user behavior, they can easily bypass basic rate-limiting and signature-based defenses, making them appealing for stealthy and persistent disruptions.

Challenges in Mitigation

HTTP flood attacks pose unique challenges because of their ability to blend in with legitimate web traffic. These attacks often use randomized request headers, realistic user agents, and behavior patterns that closely resemble human activity. Traditional DDoS defenses, which focus on identifying high traffic volumes or malformed packets, are less effective. Even behavior-based defenses struggle when bots simulate natural browsing behavior or distribute requests over time. Mitigating such attacks requires context-aware models, intelligent request scoring, and real-time traffic profiling. Machine learning-based techniques are promising but must operate under strict performance constraints, especially when deployed at or near the network edge. [40].

Practical Network Considerations

In real-world scenarios, attack and legitimate traffic are typically routed through the same ingress interface of a network router. If the total incoming traffic remains below the router's egress bandwidth (e.g., 1 Gbps Ethernet), the flood may have minimal impact. However, if the aggregate volume exceeds that limit, the router begins queuing packets on its outgoing interface. Once the queue is full, the router discards excess packets—regardless of whether they are malicious or legitimate. Without intelligent filtering in place, critical requests from real users may be dropped, resulting in service degradation or denial. Even with machine learning-based classifiers, computational overhead may prevent real-time decision-making under high-load conditions, necessitating assistance from upstream ISPs or cloud-based filtering solutions.

Model Overview

Figure 2.1 provides a high-level illustration of an HTTP flood attack. While simplified, it highlights the flow of traffic from botnets to the server via a router. In practice, such deployments involve complex routing paths, multi-homed interfaces, and real-time traffic queuing. Accurate modeling of these flows is essential for understanding the limitations of detection systems and the need for scalable, adaptive mitigation strategies [1].

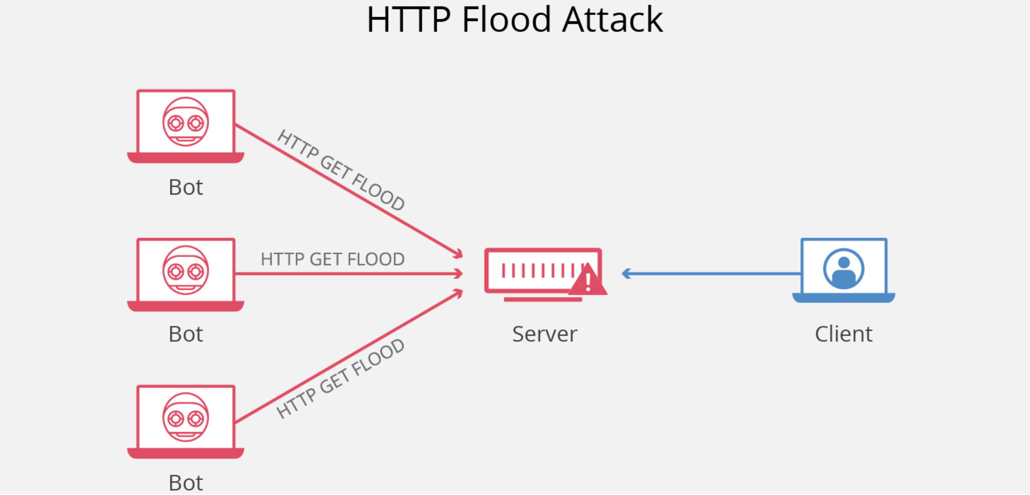

Figure 2.1: Illustration of an HTTP Flood Attack [1].

2.2 Machine Learning and Deep Learning

2.2.1 Machine Learning

Machine learning (ML) has become one of the most transformative technologies in recent years, reshaping industries by enabling systems to learn and adapt without explicit programming. At its core, ML involves training algorithms on data to identify patterns, make decisions, and improve over time. This adaptability makes ML particularly valuable in dynamic and rapidly evolving domains such as cybersecurity.

Types of Machine Learning:

1. **Supervised Learning**:
   Supervised learning involves training a model on a labeled dataset, where the inputs and their corresponding outputs are known. The algorithm learns to map the input to the output and generalize this knowledge to unseen data [40].

   - **Applications**: Classification tasks like spam detection or regression tasks like predicting network traffic volumes.
   - **Examples**: Decision trees, linear regression, support vector machines (SVM).

2. **Unsupervised Learning**:
   In unsupervised learning, the model works with unlabeled data to identify hidden patterns or structures [40].

   - **Applications**: Anomaly detection in network traffic, customer segmentation.
   - **Examples**: Clustering algorithms like k-means or hierarchical clustering.

3. **Reinforcement Learning**:
   Reinforcement learning involves training an agent to make a series of decisions by interacting with an environment. The agent receives rewards for desirable actions and penalties for undesirable ones, learning to optimize its actions over time [40] .

   - **Applications**: Optimizing network defenses, autonomous systems.
   - **Examples**: Q-learning, deep Q-networks.

Benefits in Cybersecurity: ML enhances threat detection by identifying patterns and anomalies in large datasets, allowing for early detection of sophisticated attacks such as DDoS. It also enables predictive analysis, helping organizations anticipate and mitigate potential threats before they occur.

Libraries Used in Machine Learning: Machine learning is supported by a wide range of libraries and frameworks that facilitate the development, training, and deployment of models. These libraries offer tools for data manipulation, model implementation, visualization, and utility functions that streamline the machine learning workflow. For instance, `pandas` and `numpy` are essential for efficient data handling and transformation, allowing users to preprocess large datasets effectively. Visualization libraries such as `seaborn` and `matplotlib` enable researchers to create detailed plots and graphs, which are crucial for understanding data distributions and model performance. Frameworks like `scikit-learn` provide a comprehensive suite of machine learning algorithms, including support vector machines (SVM), logistic regression, and random forests, making it easier to experiment with various models. Additionally, utility libraries such as `pickle` are used to save and load trained models, ensuring that they can be reused without retraining. Together, these libraries form the backbone of any machine learning project, enabling both flexibility and efficiency.

2.2.2 Deep Learning

Deep learning (DL), a subset of machine learning, uses neural networks with multiple layers to process and analyze large amounts of data. DL excels at identifying complex patterns in unstructured data such as images, audio, and text, making it highly effective for detecting sophisticated cyberattacks [42].

Key Components of Deep Learning:

- **Neural Networks**: Networks consist of layers of interconnected nodes (neurons), each performing computations to extract features from the data.

- **Activation Functions**: These introduce non-linearities into the model, enabling it to learn complex patterns. Examples include ReLU, sigmoid, and tanh.

- **Optimization Algorithms**: Algorithms like stochastic gradient descent (SGD) adjust the weights of the network to minimize the error during training.

Applications in Cybersecurity: DL models can analyze network traffic patterns to detect anomalies indicative of DDoS attacks. They are particularly effective in identifying complex, multi-dimensional attack signatures that traditional methods might miss.

Libraries Used in Deep Learning: Deep learning also relies on a comprehensive suite of libraries and frameworks designed to handle the complexities of neural networks and large-scale data processing. One of the most prominent frameworks is `tensorflow.keras`, which allows developers to build and train deep neural networks efficiently. TensorFlow also provides additional tools such as TensorBoard, enabling users to visualize training progress, model performance, and network architecture. For large-scale experiments, TensorFlow facilitates distributed training, making it a go-to library for deep learning projects. Visualization tools like Matplotlib are also employed to plot metrics, visualize feature maps, or understand layer activations in a deep learning pipeline. Together, these libraries provide the flexibility and computational power needed to explore complex relationships in datasets while improving model accuracy and scalability.

Deep learning's ability to process large-scale data and its adaptability to evolving threats make it an indispensable tool in modern cybersecurity defenses.

2.2.3 Algorithms Used

The detection of HTTP flood attacks is facilitated by employing a diverse set of machine learning and deep learning models. These models are designed to identify patterns, classify data, and detect anomalies in network traffic effectively. By leveraging these techniques, this research aims to enhance the accuracy and reliability of detecting malicious traffic patterns indicative of HTTP flood attacks. The models operate on preprocessed network traffic data, learning to differentiate between normal and attack traffic based on historical patterns and features. Their ability to generalize from training data to unseen traffic makes them highly effective in real-world applications.

This section provides an overview of the machine learning and deep learning algorithms utilized in the research, detailing their methodologies, relevance to the problem domain, and anticipated performance in detecting HTTP flood attacks.

Machine Learning Algorithms

Machine learning algorithms play a crucial role in this research by providing the foundational tools for classifying and detecting anomalies in network traffic. The following algorithms were applied to construct and train models capable of recognizing HTTP flood attack patterns:

1. **Linear Support Vector Machine (SVM)**:
   Linear SVM is a supervised learning algorithm that identifies the optimal hyperplane to separate data into different classes. It is particularly effective for binary classification tasks, such as distinguishing between normal and malicious traffic. By maximizing the margin between data points of different classes, Linear SVM ensures accurate classification even in high-dimensional spaces. Its simplicity and efficiency make it a widely used algorithm in cybersecurity applications [20].

2. **LightGBM (LGB)**:
   LightGBM is a gradient boosting framework optimized for speed and performance. It builds decision trees sequentially, reducing errors from previous iterations. LightGBM excels at handling large, high-dimensional datasets, which are common in network traffic analysis. Its ability to capture complex interactions between features makes it a powerful tool for detecting nuanced patterns in HTTP flood attacks [9].

3. **K-Nearest Neighbors (KNN)**:
   KNN is a simple, instance-based learning algorithm that classifies data points based on their proximity to labeled examples. In the context of HTTP flood attacks, KNN identifies malicious traffic by comparing it to known attack patterns. While effective for smaller datasets, its performance may degrade with larger data volumes, requiring optimization for high-speed classification [74].

4. **Random Forest (RF)**:
   Random Forest is an ensemble learning technique that combines multiple decision trees to improve classification accuracy and reduce overfitting. By aggregating the predictions of individual trees, Random Forest provides a robust model capable of handling noisy and imbalanced datasets. It is particularly effective in identifying subtle patterns in network traffic indicative of HTTP flood attacks [48].

5. **Logistic Regression**:
   Logistic Regression is a statistical model commonly used for binary classification. Despite its simplicity, it serves as a strong baseline algorithm for detecting DDoS attacks. Logistic Regression models the relationship between input features and the probability of belonging to a particular class, making it useful for initial assessments of network traffic data [75].

Expected Performance: Each of these algorithms offers unique advantages, from interpretability to scalability, enabling a comprehensive evaluation of their effectiveness in detecting HTTP flood attacks. Their combined use allows for a comparative analysis of performance, leading to the identification of the most suitable model for the task.

Deep Learning Algorithms

Deep learning algorithms provide the capability to model complex, nonlinear relationships in data, making them ideal for tasks requiring high accuracy and adaptability. The following deep learning architectures are employed in this research:

1. **Convolutional Neural Networks (CNNs)**:
   CNNs are primarily designed for spatial data processing, such as images, but have been adapted for analyzing structured data like network logs. They use convolutional layers to extract features, pooling layers to reduce dimensionality, and fully connected layers to classify data. CNNs are particularly effective in detecting HTTP flood attacks by analyzing packet-level data and identifying patterns indicative of malicious activity [58].

2. **Long Short-Term Memory Networks (LSTMs)**:
   LSTMs are a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. Network traffic data, which is inherently temporal, is well-suited for LSTM analysis. By retaining information across time steps, LSTMs detect anomalies in traffic patterns that may indicate an ongoing HTTP flood attack. Their ability to handle varying sequence lengths makes them highly versatile in this application [58].

Expected Impact: CNNs and LSTMs complement each other in this research. While CNNs excel at extracting spatial features from structured data, LSTMs are adept at recognizing temporal patterns in sequential data. Together, they provide a holistic approach to analyzing network traffic, improving the detection of HTTP flood attacks.

Relevance to HTTP Flood Attacks

HTTP flood attacks represent a sophisticated type of DDoS attack targeting the application layer. These attacks mimic legitimate traffic, making them challenging to detect with traditional rule-based methods. The algorithms described above are selected for their ability to handle such challenges:

- Machine learning models like SVM, Random Forest, and Logistic Regression are effective for quick, interpretable classifications of network traffic. They serve as reliable tools for identifying deviations from normal traffic patterns.

- Deep learning architectures like CNNs and LSTMs are better suited for detecting complex, high-dimensional patterns in HTTP flood attacks. Their ability to analyze packet-level and temporal data provides a significant advantage in identifying subtle indicators of malicious activity.

By leveraging these algorithms, this research aims to develop a robust detection framework capable of identifying HTTP flood attacks with high accuracy and reliability.

2.2.4 Model Operations

The operation of machine learning and deep learning models plays a critical role in ensuring their effectiveness in detecting HTTP flood attacks. This section outlines the essential steps involved in model operations, including data preprocessing, training, validation, testing, and evaluation.

Data Preprocessing

Data preprocessing is a crucial step in machine learning and deep learning workflows. Raw network traffic data is often unstructured, noisy, or incomplete, which can negatively impact model performance. Preprocessing ensures that data is clean, consistent, and suitable for analysis. Key preprocessing steps include:

- **Data Cleaning**: Removing or imputing missing values, filtering outliers, and correcting inconsistencies in the dataset.
- **Feature Engineering**: Extracting relevant features from raw traffic logs, such as packet size, time intervals, and HTTP request types. These features provide the model with meaningful input to learn from.
- **Normalization and Scaling**: Transforming features to a uniform scale to ensure that no single feature disproportionately influences the model. For example, packet sizes may be scaled to values between 0 and 1 using min-max scaling.
- **Encoding Categorical Variables**: Converting categorical data, such as protocol types, into numerical representations using techniques like one-hot encoding or label encoding.

Model Training

Training involves teaching the model to recognize patterns in the data by adjusting its parameters to minimize error. During training:

- The dataset is split into training, validation, and testing subsets in a ratio of 60%, 20%, and 20%, respectively.
- Models are fed labeled data (in the case of supervised learning) or unlabeled data (in unsupervised learning) to identify patterns or classify traffic as normal or malicious.
- Optimization algorithms, such as gradient descent, adjust the model's parameters to minimize the loss function, which measures the difference between predicted and actual outputs.

Validation and Hyperparameter Tuning

Validation is essential for assessing the model's performance on unseen data and preventing overfitting. During validation:

- Hyperparameters, such as learning rates, tree depths (for decision tree models), or the number of layers and neurons (for deep learning models), are tuned to optimize performance.
- Techniques such as k-fold cross-validation are used to evaluate the model on multiple subsets of the data, ensuring robust performance across different samples.

Testing and Evaluation

Testing evaluates the model's generalization capability by measuring its performance on previously unseen data. Key evaluation metrics include:

- **Accuracy**: The proportion of correctly classified instances over the total instances.
- **Precision**: The ratio of true positives to the sum of true positives and false positives, indicating the model's ability to avoid false alarms.
- **Recall (Sensitivity)**: The ratio of true positives to the sum of true positives and false negatives, measuring the model's ability to detect malicious traffic.
- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of model performance.
- **ROC-AUC (Receiver Operating Characteristic - Area Under Curve)**: Evaluates the trade-off between true positive and false positive rates.

Challenges in Model Operations

Despite their effectiveness, machine learning and deep learning models face several operational challenges:

- **Data Imbalance**: In network traffic datasets, malicious traffic often constitutes a small fraction of the data, which can lead to biased models. Techniques like oversampling or synthetic data generation (e.g., SMOTE) are used to address this issue.
- **Scalability**: Models must handle large-scale, high-velocity traffic data without compromising accuracy or speed.
- **Adversarial Attacks**: Attackers may intentionally craft inputs to deceive the model, highlighting the need for robust and secure model design.
- **Resource Constraints**: Deep learning models, in particular, require substantial computational resources for training and inference, necessitating the use of optimized algorithms and hardware acceleration (e.g., GPUs).

2.2.5 Explainable AI (XAI)

As machine learning and deep learning models become increasingly complex, understanding and interpreting their decision-making processes becomes critical. Explainable AI (XAI) addresses this challenge by providing tools and techniques that make AI systems transparent, interpretable, and trustworthy. In the context of cybersecurity, particularly HTTP flood attack detection, XAI plays a pivotal role in validating model decisions, identifying biases, and building trust among stakeholders [31].

Importance of Explainable AI

Traditional machine learning and deep learning models often operate as "black boxes," meaning their internal workings are opaque to users. While these models deliver high performance, their lack of interpretability can hinder adoption in critical domains like cybersecurity. Explainable AI bridges this gap by:

- **Enhancing Trust**: Providing insights into how and why models make certain predictions builds confidence among users, particularly in high-stakes environments.
- **Identifying Biases**: XAI helps uncover potential biases in models, ensuring fair and ethical decision-making.
- **Facilitating Debugging**: Understanding model behavior aids in identifying errors or areas for improvement, enabling more robust implementations.
- **Meeting Regulatory Requirements**: In some industries, interpretability is a legal or ethical requirement for deploying AI systems.

Tools and Techniques in XAI

In this research, **SHAP (Shapley Additive Explanations)** is the primary tool used for enhancing interpretability. SHAP assigns a Shapley value to each feature, quantifying its contribution to the prediction. This approach allows for:

- Visualizing feature importance to understand which attributes drive model decisions.
- Explaining individual predictions, enabling the identification of anomalous or unexpected classifications.
- Comparing feature impacts across multiple instances to validate model behavior.

While other techniques, such as **LIME (Local Interpretable Model-agnostic Explanations)** and feature importance analysis provided by frameworks like `scikit-learn` and `LightGBM`, are widely used for model interpretability, this research exclusively leverages SHAP due to its comprehensive capabilities and suitability for explaining machine learning and deep learning models in cybersecurity contexts.

Application of XAI in HTTP Flood Attack Detection

In this research, XAI techniques are applied to interpret the predictions of machine learning and deep learning models, ensuring transparency and accountability. Specific use cases include:

- **Explaining Detection Results**: SHAP is used to identify which features (e.g., packet size, time intervals, request types) contributed most significantly to classifying traffic as malicious or normal.
- **Validating Model Decisions**: By visualizing feature impacts, XAI techniques validate whether the model is focusing on relevant attributes, ensuring alignment with domain knowledge.
- **Enhancing Model Performance**: Insights gained from XAI tools are used to refine feature engineering processes and improve model accuracy.

Challenges and Future Directions

While XAI offers numerous benefits, it also presents challenges:

- **Scalability**: Explaining predictions for large datasets or complex models can be computationally expensive.
- **Subjectivity**: Interpretations provided by XAI tools may vary depending on the technique used, requiring careful validation.
- **Integration**: Seamlessly integrating XAI techniques into existing cybersecurity frameworks remains an area for improvement.

Future research could focus on developing more efficient XAI methods tailored to high-velocity network traffic data, ensuring real-time interpretability without compromising performance.

Chapter 3

Related work

Distributed Denial of Service (DDoS) attacks have been a persistent and evolving threat in the cybersecurity landscape, affecting organizations and individuals globally. Among the many types of DDoS attacks, HTTP flood attacks have garnered significant attention due to their ability to mimic legitimate user behavior while targeting the application layer of network infrastructure. These attacks exploit the HTTP protocol by sending an overwhelming number of GET or POST requests, thereby exhausting server resources and rendering services inaccessible to genuine users.

The increasing reliance on web-based applications has amplified the risk and impact of HTTP flood attacks. Unlike traditional DDoS attacks, which primarily target the network or transport layers, HTTP flood attacks operate at the application layer, making them harder to detect and mitigate. These attacks often bypass conventional defense mechanisms like firewalls and intrusion detection systems due to their legitimate appearance.

Research in the field has focused on developing robust methods to detect and mitigate HTTP flood attacks, employing both traditional approaches and advanced machine learning (ML) and deep learning (DL) techniques. This chapter reviews the existing literature, categorizing related work into key areas such as DDoS detection techniques, machine learning-based detection, deep learning approaches, and the integration of explainable AI (XAI) to enhance transparency and trust in detection systems. The chapter aims to highlight the strengths and limitations of current approaches while identifying gaps that this research seeks to address.

3.1 Literature Review

To provide a comprehensive understanding of the existing research landscape, a systematic literature review (SLR) was conducted. The SLR methodology ensures a rigorous and unbiased synthesis of the current state of knowledge, focusing on the detection of HTTP flood attacks. By systematically identifying, evaluating, and synthesizing relevant studies, this review aims to address key research questions and establish a foundation for the subsequent development of advanced detection techniques.

3.1.1 Research Questions for the SLR

To guide the SLR process and ensure that it addresses critical aspects of HTTP flood attack detection, the following research questions were formulated:

- **RQ1**: What are the existing methods and approaches for detecting HTTP flood attacks?
- **RQ2**: What are the strengths, weaknesses, and accuracies of existing detection techniques, particularly those using machine learning and deep learning?
- **RQ3**: What are the key performance metrics used to evaluate HTTP flood detection techniques, and how do they compare across different studies?

These research questions aim to provide a structured approach to the review, ensuring comprehensive coverage of the topic and identifying areas for future research.

3.1.2 Search Strategy

The search strategy for the SLR was designed to identify relevant studies comprehensively while maintaining a systematic and unbiased approach. The key steps in the search process included:

Databases Searched: The following digital libraries and databases were used to retrieve relevant studies:

- IEEE Xplore
- ACM Digital Library
- SpringerLink
- ScienceDirect
- Google Scholar

Search Terms: A combination of keywords and Boolean operators was used to refine the search. Example search strings include:

- `"HTTP flood attack detection"`
- `"DDoS application layer attacks AND machine learning"`
- `"Explainable AI in cybersecurity"`
- `"Deep learning for DDoS detection"`

Inclusion and Exclusion Criteria:

- **Inclusion Criteria**:
  - Studies published in peer-reviewed journals or conferences.
  - Studies focusing on HTTP flood attack detection or related application-layer attacks.
  - Studies utilizing machine learning, deep learning, or explainable AI techniques.
- **Exclusion Criteria**:
  - Studies not written in English.
  - Studies lacking experimental evaluation or comprehensive analysis.
  - Studies focusing solely on network/transport-layer attacks without relevance to HTTP flood attacks.

3.1.3 Data Extraction

The data extraction process involves systematically collecting and organizing relevant information from the selected studies to address the research questions. This process helps in synthesizing the findings and identifying patterns or gaps in the existing literature.

Data Extraction Steps:

1. **Create a Data Extraction Form**: Develop a template to collect key information from each study, such as:
   - Study title, authors, year of publication.
   - Research questions or hypotheses.
   - Methodology (e.g., machine learning models used, datasets, evaluation metrics).
   - Key findings and results.
   - Limitations and future research directions.
2. **Extract Data from Selected Studies**: Populate the data extraction form with information from each selected study.
3. **Organize Extracted Data into Tables**: Use tables to summarize key findings, such as performance metrics of different machine learning models, benefits of XAI, or effectiveness of anomaly detection techniques.
4. **Analyze Extracted Data**: Synthesize the findings to identify trends, strengths, weaknesses, and gaps in the literature.

The systematic data extraction process ensures that the findings are accurately documented and effectively inform the subsequent stages of this research.

3.1.4 Data synthesis

The data synthesis process aims to consolidate insights derived from the reviewed studies, emphasizing the methods, strengths, and weaknesses identified in the tables. By analyzing these findings, this section provides a comprehensive perspective on the current state of research in HTTP flood attack detection using machine learning and deep learning techniques. Additionally, the performance metrics and experimental setups are examined to understand trends and identify gaps relevant to our research objectives.

The synthesis of Table 3.1 highlights the methodologies employed, including advanced ensemble techniques and deep learning models. These studies demonstrate diverse preprocessing strategies and hyperparameter tuning methods aimed at optimizing model performance. While most approaches exhibit high detection accuracy, such as DDoS-Net's 99% accuracy and LSTM-based models achieving 97.19%, certain limitations, including outdated datasets and the lack of real-time feasibility analysis, are prevalent. These limitations underline the need for updated datasets and more practical implementations for HTTP flood attack detection.

Similarly, Table 3.2 focuses on performance metrics, including accuracy, recall, and precision, across various machine learning and deep learning algorithms. Models like CNN-LSTM and feature hashing demonstrate exceptional precision (98.42%) and recall (97.6%), showcasing their effectiveness in detecting attacks. However, significant challenges, such as imbalanced datasets and high computational requirements for training, hinder their scalability. For instance, the training time for certain models exceeds 15,000 seconds, which may not be feasible for real-world scenarios requiring rapid deployment and evaluation.

Table 3.1: Methodologies, strengths, and weaknesses of different studies using ML/DL for DDoS detection.

Reference	**Methodology**	**Strengths**	**Weaknesses**
[56]	Utilized RF with Non-Symmetric Deep Auto Encoder	Presented a non-symmetric deep AE and RF classifier-based DDoS detection, which lowers the model complexity	The model was examined with outdated datasets, and its performance on the minority R2L and U2R classes was on the low side.
[7]	Employed a Particle Swarm Algorithm and a Fast Learning Network	The suggested model outperformed other FLN-based models using several additional optimization strategies	Outdated dataset used. Additionally, fewer training data led to a lower detection rate.

Reference	Methodology	Strengths	Weaknesses
[67]	Utilized SVM and a Sparse Auto Encoder	SVM was effectively used as a classifier with SSAE for feature extraction to identify DDoS attacks	The model was evaluated using an outdated dataset; while the model's detection rates for U2R and R2L attacks were respectable, they were lower than for the other attack classes in the dataset.
[41]	Used SVM and Deep Belief Network	DBN was utilized to extract features, which were then passed on to an ensemble SVM before being predicted through a voting method	Model complexity and training require a longer time with more deep layers.
[72]	K-Means Clustering and Random Forest-based multilevel model	The clustering idea was applied in combination with RF to offer a multilayer intrusion detection model; the model performed better than average in identifying assaults	Used the outdated KDDCup99 dataset to test the model.
[52]	Two ML models (DNN and LSTM) were proposed for the detection of DDoS assaults	The performance of these models significantly improved, with DNN and LSTM accuracy levels of 98.72% and 96.15%, respectively. AUC values for the DNN and LSTM were 0.987 and 0.989, respectively	The authors did not use real-time detection, and only binary class classification was carried out.

Reference	Methodology	Strengths	Weaknesses
[17]	Suggested a Multi-channel CNN architecture for DDoS assaults	The MCCNN performed better on the constrained dataset	The outcomes of multi-class and single-class classification models were not significantly different; the complexity of the multi-class model makes it unsuitable for validation in real-time circumstances.
[54]	Used two datasets to test the suggested model against classification algorithms including DT, SVM, KNN, and NN	Suggested model scored well and provided 99% accuracy on both datasets	In this method, the data were transformed into a matrix by extending one column. This may influence how the model learns.
[27]	Suggested a deep CNN framework for the detection of DDoS assaults in SDN	Ensemble CNN technique performed better than the currently used rival approaches, attaining a collective 99.45% accuracy rate	This strategy requires longer training and testing periods. As a result, the mitigating mechanism may be impacted, meaning that assaults can cause more damage.
[64]	Suggested an information entropy and DL technique to identify DDoS assaults in an SDN context	The CNN outperformed alternatives in terms of precision, accuracy, F1-score, and recall, with an accuracy rate of 98.98%	The model required a longer time to perform detection.
[35]	Created a CNN-based model to identify DoS attacks	The CNN model is better able to recognize unique DoS assaults with similar features. Additionally, CNN kernel size had no discernible effect on either binary or multiclass categorization	Longer time detection.
[36]	Suggested a deep learning approach	DDoS assault detection was 98% accurate	Long time required for detection.

Reference	Methodology	Strengths	Weaknesses
[49]	A DL-based approach was developed	For all the test cases, it was noted that the LSTM model displayed 98.88% accuracy. The module was able to prevent an attacked packet from reaching the cloud server via the SDN OF switch	Only DDoS assaults at the network or transport level were examined; real-time feasibility analyses of the proposed model were not conducted.
[38]	A four-layered architectural model with two LSTM layers was suggested	Experimental findings demonstrated that the LSTM-based technique performed better than other approaches	When the flow consisted of short packets, N was padded with fictitious packets. These padding settings have the potential to degrade performance and impact how the suggested model learns.
[59]	Two methods, a hybrid-based IDS and a DL model based on LSTM, were proposed for DoS/DDoS assaults detection	The LSTM-based model reached an accuracy of 99.19%	A long time was required for detection.
[16]	Combined a deep ANN and AE model	The best F1 values with the ReLu activation function were obtained (0.8985). For the activation functions of softplus, softsign, relu, and tanh, the overall accuracy, and precision are close to 99%	The activation functions are the sole thing this article focuses on

Reference	Methodology	Strengths	Weaknesses
[70]	A five-layered AE model was developed for efficient unsupervised DDoS detection	AE-D3F achieved nearly 100% DR with less than 0.5% FPR, although the RE threshold value must be specified. This method compensates for the lack of labeled attack data by training the model using only regular traffic	Used an outdated dataset
[24]	Combined AE and RNN to produce DDoS-Net for identifying DDoS assaults in SD	The results indicated that DDoS-Net performed better than six traditional ML techniques (DT, NB, RF, SVM, Booster, and LR) in terms of accuracy, recall, precision, and F1-score. The proposed method obtained 99% accuracy and an AUC of 98.8	The dataset used offline analysis, and multiclass classification was not carried out
[46]	DL-based strategy was proposed to identify sluggish DDoS assaults in SDNs using a CNN–LSTM model	The suggested model performed better than other approaches, obtaining more than 99.5 percent across all performance criteria	The dataset used offline analysis
[18]	Suggested a minimax gradient-based deep reinforcement learning technique	When compared to cutting-edge algorithms, the suggested policy-based GPDS algorithm outperformed them in terms of anti-jamming performance	

Table 3.2: The most recent studies on DDoS attack detection using ML/DL

Reference	Preprocessing Strategies	Hyperparameter Values	Experiment Setup	Performance Metrics
[45]		CNN, AE, and RNN		Accuracy for DCNN and LSTM of 85% and 89%, respectively
[32]		Six different ML models		Accuracy between 4.01% and 30.59%
[25]		Ensemble learning		Accuracy = 85.2%, Precision = 86.5%, Recall = 85.2%, F1-score = 84.9%
[50]		A 1D CNN layer with ReLu function, LSTM layer with Adam optimizer, dropout layer with a rate of 0.5, FC layer, and dense layer with sigmoid function make up the CNN–LSTM model	Intel Core-i7 with Keras, TensorFlow, and MATLAB	Precision = 97.41%, Recall = 99.1%, Accuracy = 97.36%
[43]	Feature hashing and BOW	Two hidden FC layers of 256 neurons each with ReLU activation function and one neuron with sigmoid activation function make up the LSTM module		Recall = 97.6%, Accuracy = 98.15%, Precision = 98.42%, TNR = 98.4%, FPR = 1.6%, F1-Score = 98.05%

Reference	**Preprocessing Strategies**	**Hyperparameter Values**	**Experiment Setup**	**Performance Metrics**
[51]	Min-max	Maxpooling, LSTM, and dropout layers with Relu activation function after a 1D CNN; learning rate = 0.001, batch size = 256, epochs = 100, and dropout rate = 0.2%.	NVIDIA Tesla VIOO GPUs with TensorFlow and Keras software	Accuracy = 99.03%, recall = 99.35%, Precision = 99.26%, F1-score = 99.36%, Training Time = 15,313.10 s
[55]	Random subspace method	Population size: 20, Loudness: 0.25, Pulse rate: 0.5, Number of iterations: 200	KDD99, NSL, Kyoto datasets, Matlab R2012a, 2.4 GHz CPU, 32GB RAM	Accuracy (KDD99): 98.91%, Accuracy (NSL): 97.49%, Accuracy (Kyoto): 99.20%
[57]	Non-symmetric Deep Autoencoder (NDAE) with Random Forest	Three hidden layers for each NDAE, activation function: sigmoid, cross-validation used	GPU-enabled TensorFlow, Intel Xeon 3.60GHz, 16GB RAM, NVIDIA GTX 750 GPU, KDD Cup '99, NSL-KDD datasets	Accuracy (KDD99): 97.85%, Accuracy (NSL-KDD): 85.42%, F1-score (NSL-KDD): 87.37%

Reference	Preprocessing Strategies	Hyperparameter Values	Experiment Setup	Performance Metrics
[53]	Data normalization, Z-score normalization	First dense/recurrent layer with 60 neurons and ReLU activation, dropout = 0.2, another fully connected layer with the same configuration	GPU NVIDIA Quadro K2200, Intel(R) Xenon CPU E5-2630 v3@ 2.40GHz, 64-bit Windows 10 Pro 1809	Accuracy (DNN, Data 1): 99.59%, Accuracy (LSTM, Data 1): 99.20%, Evaluation accuracy (DNN, unknown attacks): 88.29%, Evaluation accuracy (LSTM, unknown attacks): 87.64%
[63]	Raw socket programming, feature engineering with PCA	Naive Bayes, Random Forest, Decision Tree, and Deep Neural Network (DNN)	OpenStack-based private cloud with Ubuntu machines	Accuracy (DNN, cloud dataset): 96%, Accuracy (Decision Tree, cloud dataset): 92%, Accuracy (KNN, cloud dataset): 91%
[44]	Preprocessing involved removal of flowID, timestamps, and IP addresses from CICIDS2017 dataset	Activation Function: ReLU, Loss Function: Categorical Cross Entropy, Optimizer: Adam, Hidden Layers: 4, Patience Value: 5, Epochs: 43	CICIDS2017 dataset, implemented using Keras and SciKit on 2.4 GHz CPU	Accuracy: 99.61%, Precision: 99.1%, Recall: 99.1%, F1-Score: 99.1%

Reference	**Preprocessing Strategies**	**Hyperparameter Values**	**Experiment Setup**	**Performance Metrics**
[12]	Dataset preprocessing including removal of categorical fields, normalization using Min-Max scaling, handling imbalance through cost-sensitive learning	Seven hidden layers, Activation Function: ReLU, Loss Function: Cross Entropy, Optimizer: Adam, Dropout: 0.2, Learning Rate: 0.001	CICIDS2017 dataset, trained on NVIDIA Tesla K80 GPU with 24 GB GDDR5 Memory	F1-Score (Validation): 0.9866, Training Loss: 0.0583, Validation Loss: 0.0402, Accuracy: 98.66%
[29]	Preprocessing using feature engineering with Random Forest importance and Principal Component Analysis (PCA)	Layers: Fully Connected Dense Layers (2 hidden layers with 128 neurons each), Activation: Sigmoid, Optimizer: RMSProp, Learning Rate: 0.0001	UNSW-NB15 dataset, NVIDIA GeForce GTX 1080 GPU, TensorFlow backend	Accuracy: 97.52%, Precision: 96.74%, Recall: 96.40%, F1-Score: 96.57%
[19]	Removal of missing data, normalization using Z-score, oversampling minority classes using SMOTE	CNN with three convolutional layers, Activation: ReLU, Kernel Size: 3x3, Pooling: MaxPooling, Dense Layer with Dropout = 0.25, Batch Size: 128	Trained using CICIDS2017 dataset on NVIDIA Tesla P100 GPU with 16 GB Memory	Accuracy: 98.94%, Precision: 98.23%, Recall: 97.88%, F1-Score: 98.05%
[10]	Feature selection using Pearson Correlation Coefficient, handling missing values by mean imputation	1D-CNN: Layers: Convolutional Layers (64 filters), Activation Function: ReLU, Optimizer: Adam, Learning Rate: 0.001	NSL-KDD and CICIDS2017 datasets, executed on Intel Core i7 and 16GB RAM	Accuracy (NSL-KDD): 96.82%, Accuracy (CICIDS2017): 97.45%, F1-Score: 96.23%

Reference	**Preprocessing Strategies**	**Hyperparameter Values**	**Experiment Setup**	**Performance Metrics**
[21]	Feature selection using Chi-square test and Recursive Feature Elimination (RFE)	SVM classifier with RBF kernel, Regularization parameter: C=1, Gamma=0.01	NSL-KDD and CI-CIDS2017 datasets, Experimented on Intel Core i5, 16GB RAM	Accuracy (NSL-KDD): 96.88%, Precision: 96.42%, Recall: 95.73%, F1-Score: 96.07%
[65]	Preprocessing with SMOTE for balancing classes, StandardScaler for normalization	Dense Neural Network with two hidden layers (128 and 64 neurons), Activation: ReLU, Optimizer: Adam, Batch Size: 64	UNSW-NB15 dataset, TensorFlow 2.0, Executed on NVIDIA Tesla P100 GPU	Accuracy: 95.72%, Precision: 95.01%, Recall: 94.87%, F1-Score: 94.94%
[22]	Data normalization and feature reduction techniques	Batch size: 100, Learning rate: 0.001, Epochs: 50	CICDDoS2019 dataset, ResNet architecture, 2.4 GHz CPU, 32GB RAM	Accuracy: 99.99%, Precision: 87%, F1-Score: 87%
[30]	Feature transformation and time-window segmentation	Window size: 500, Stride: 5, Layers: 3LSTM, Batch size: 64, Learning rate: 0.0001	ISCX2012 dataset, Keras framework, 2 NVIDIA K80 GPUs, Ubuntu 14.04 OS	Accuracy: 99.88%, F1-Score: 99.79%
[37]	Data packet processing and bidirectional RNN feature extraction	Convolution kernel: 5, Stride: 1, Activation: tanh, Layers: Bidirectional RNN	OpenFlow-based SDN, ISCX dataset, Keras framework	Detection accuracy: 99.6%

Reference	**Preprocessing Strategies**	**Hyperparameter Values**	**Experiment Setup**	**Performance Metrics**
[39]	Min-Max scaling	Learning rate: 0.0005, Dropout rate: 0.3, Filter size: 64, Kernel size: 5	Generated datasets using SDN switches; Benign and slow DDoS attack flows; Python implementation; Training data: 80%, Validation: 20%	Accuracy: 99.998%, Precision: 99.989%, Specificity: 99.997%, Recall: 100%, F1 Score: 99.994%
[8]	Flow feature extraction using single IP flow records	GRU layer: 32 cells, Dropout rate: 0.5, Fully connected layer: 10 neurons	Tested on CICDDoS 2019 and CICIDS 2018 datasets; Real IP Flow data; Training environment: Intel Core i7, 32GB RAM	Detection rate: High; Precision: High; Ability to analyze large flows per second confirmed; Specific performance metrics not stated
[47]	Feature selection using Random Forest importance scores, Normalization using Z-score scaling	Optimizer: SGD, Learning rate: 0.01, Batch size: 64, CNN with 2 convolutional layers, Activation: ReLU	CICIDS2017 dataset, Training on NVIDIA GTX 1080 GPU	Accuracy: 97.89%, Precision: 97.45%, Recall: 97.68%, F1-Score: 97.56%
[13]	Raw feature extraction, Packet-level inspection using custom scripts	GRU Layer: 64 cells, Fully connected Dense Layer with 32 neurons, Activation: Sigmoid	ISCX2012 and KDD99 datasets, Implemented in Python on Intel Core i5, 16 GB RAM	Detection Rate: 99.1%, False Positive Rate: 0.02%, Precision: 99.3%, F1-Score: 99.2%

Reference	**Preprocessing Strategies**	**Hyperparameter Values**	**Experiment Setup**	**Performance Metrics**
[14]	Data augmentation using SMOTE and Min-Max normalization	Learning rate: 0.0001, Dropout rate: 0.4, Optimizer: Adam, 2-layer LSTM with 256 units each	NSL-KDD dataset, Training on NVIDIA RTX 2080 Ti GPU, TensorFlow/Keras backend	Accuracy: 98.5%, Precision: 97.4%, Recall: 97.8%, F1-Score: 97.6%
[33]	Traffic normalization, Dimensionality reduction using PCA	Random Forest: Number of Trees: 100, Max Depth: 20; Naive Bayes with Gaussian Distribution	UNSW-NB15 dataset, Training on Intel Xeon E5-2620 CPU with 32 GB RAM	Accuracy: 96.8%, Precision: 94.5%, Recall: 94.8%, F1-Score: 94.65%
[71]	Feature scaling using StandardScaler, Removal of redundant features	Learning rate: 0.001, Batch size: 128, Layers: 3 CNN + 2 Fully Connected Layers, Activation: ReLU	CICIDS2017 dataset, Implementation using TensorFlow, NVIDIA Tesla P100 GPU	Accuracy: 98.65%, Precision: 98.24%, Recall: 97.98%, F1-Score: 98.11%

Trends and Insights

By synthesizing the data from Tables 3.1 and 3.2, several trends and insights emerge. First, deep learning approaches, particularly hybrid models like CNN-LSTM, consistently outperform traditional machine learning techniques in terms of accuracy and recall. This trend underscores the potential of deep learning in addressing the complexities of HTTP flood attacks. Second, the reliance on synthetic datasets, such as KDD99 and NSL-KDD, remains a common practice, raising questions about the generalizability of the findings to real-world scenarios. Third, while many studies achieve high performance metrics, the lack of emphasis on real-time detection capabilities highlights a significant gap in the current research.

Discussion

The trends observed from the synthesized data indicate a clear preference for deep learning methodologies, with hybrid models such as CNN-LSTM emerging as top performers. These models not only offer higher detection accuracies but also exhibit improved recall and precision, which are critical for minimizing false negatives in DDoS detection. However, their reliance on synthetic datasets introduces concerns regarding generalizability. While synthetic datasets enable controlled experimentation, they fail to capture the complexity and variability of real-world network traffic, potentially limiting the practical applicability of the findings.

Furthermore, the lack of real-time implementation remains a critical challenge. Many studies prioritize high accuracy and precision but overlook the need for low-latency solutions that can be deployed in live network environments. This gap underscores the importance of developing lightweight, scalable models that can operate effectively in real-time conditions. Another point of concern is the computational overhead associated with training complex deep learning models, which may limit their adoption in resource-constrained environments.

Conclusion

In summary, the review of existing methodologies and performance metrics reveals significant advancements in the detection of HTTP flood attacks using ML and DL techniques. Hybrid models like CNN-LSTM demonstrate superior performance, but their applicability is constrained by reliance on synthetic datasets and computational demands. The emphasis on high accuracy metrics, often at the expense of scalability and real-time capabilities, highlights a critical area for improvement. These observations form the foundation for identifying research gaps and framing the objectives of this study.

3.2 Identified Research Gaps

Despite the advancements in using machine learning and deep learning models for detecting HTTP flood attacks, several critical gaps remain. Firstly, the existing models heavily rely on synthetic datasets such as KDD99 and NSL-KDD, which do not accurately reflect the complexities and variability of real-world HTTP traffic.

This reliance limits the generalizability and practical deployment of the proposed solutions.

Secondly, while many studies focus on achieving high accuracy and recall, the interpretability of these models remains underexplored. The "black-box" nature of many deep learning models hinders the ability to understand how decisions are made, which is critical for building trust and ensuring ethical deployment in sensitive applications. Integrating explainable AI (XAI) into these models is a promising avenue to address this gap but has not been widely implemented.

Thirdly, the computational overhead associated with training and deploying complex deep learning models presents a significant challenge. Current approaches often require high-performance computing resources, making them unsuitable for real-time applications or deployment in resource-constrained environments.

Finally, while hybrid models such as CNN-LSTM show promise, there is limited research on systematically comparing multiple ML and DL models for HTTP flood detection in a unified framework. Evaluating diverse models under consistent conditions and integrating explainability to assess their decision-making processes would provide deeper insights into their strengths and limitations.

This study aims to address these gaps by systematically comparing ML and DL models for HTTP flood detection, incorporating explainable AI techniques to enhance interpretability, and utilizing datasets that better reflect real-world traffic scenarios. By doing so, it seeks to advance the state-of-the-art in HTTP flood detection while addressing the practical challenges of deployment and scalability.

Chapter 4

Method

This chapter outlines the systematic approach employed to address the research objectives of detecting HTTP flood attacks using machine learning (ML) and deep learning (DL) techniques. It provides a comprehensive description of the implementation and experimentation processes, ensuring the reliability and reproducibility of the findings. The methodology is structured into distinct sections, beginning with the tools and technologies utilized, followed by details on setting up the experimental environment, configuring the network for attack simulation, and implementing models. The chapter concludes with a discussion on data collection, preprocessing, model evaluation, and the integration of explainable AI (XAI) to interpret the results effectively. Each section is designed to align with the research questions and ensure that the objectives are systematically addressed.

4.1 Implementation

4.1.1 Tools and Technologies

In this study, a variety of tools and technologies were utilized to implement, train, and evaluate machine learning (ML) and deep learning (DL) models for the detection of HTTP flood attacks. The selection of tools was driven by their robustness, flexibility, and community support, ensuring the methodology adhered to state-of-the-art practices while maintaining reproducibility.

Programming Environment and Libraries

The primary programming language used for this research was Python (version 3.10.2) due to its extensive ecosystem of libraries and frameworks for ML and DL tasks. Key libraries employed include:

- **Scikit-learn**: A widely used library for implementing machine learning models and preprocessing techniques. It was used for training traditional ML models such as Support Vector Machines (SVM), Random Forest (RF), and Logistic Regression (LR). The library's comprehensive set of tools for feature selection, cross-validation, and model evaluation proved invaluable.

- **TensorFlow/Keras**: These frameworks were utilized for developing and training deep learning models, including Convolutional Neural Networks (CNNs), Long Short-Term Memory networks (LSTMs), and hybrid CNN-LSTM architectures. TensorFlow's scalability and Keras' user-friendly interface allowed for efficient experimentation and model tuning.

- **SHAP (Shapley Additive Explanations)**: This library was employed for explainability purposes, enabling the interpretation of model predictions by identifying the contribution of each feature to the decision-making process.
- **Pandas and NumPy**: These libraries were used for data manipulation and analysis, providing efficient tools for handling large datasets and performing operations such as normalization and encoding.
- **Seaborn and Matplotlib**: Visualization libraries were used to generate plots and graphs for performance metrics, aiding in the analysis and presentation of results.

Hardware and Computational Resources

The experiments were conducted on an MSI Katana GF66 gaming laptop, which provided high performance for computationally demanding tasks. The detailed specifications of the hardware are as follows:

- **Processor (CPU)**: 11th Gen Intel® Core™ i7 Processor
- **Graphics Card (GPU)**: NVIDIA® GeForce RTX™ 3070 Laptop GPU with 8GB GDDR6 memory
- **Memory (RAM)**: 16GB DDR4-3200, expandable up to 64GB
- **Storage**: 1TB NVMe SSD
- **Display**: 15.6" Full HD (1920x1080) IPS-Level display with a 144Hz refresh rate
- **Operating System**: Windows 10 Home (upgraded to Windows 11)
- **Additional Peripherals**: Gigabit LAN, Wi-Fi 6, and multiple USB ports for connectivity

This configuration ensured efficient handling of large datasets, seamless execution of ML and DL models, and rapid processing during training and evaluation.

Development Tools

The following development tools and platforms facilitated the implementation and management of the project:

- **Jupyter Notebooks**: Version 6.27.1, used for writing and testing code interactively, allowing for iterative development and visualization of results.
- **Anaconda**: A Python distribution that streamlined the installation and management of libraries and dependencies.

- **Version Control (Git)**: Git and GitHub were used for version control, ensuring the project remained organized and changes were tracked systematically.

Why these tools were chosen because The selection of these tools and technologies was based on their ability to handle the specific requirements of the study. For instance, TensorFlow and Keras were chosen for their efficiency in implementing and training complex DL models, while Scikit-learn's extensive suite of ML algorithms made it ideal for comparative analysis. Additionally, SHAP's focus on explainability aligned with the project's objective to provide interpretable results. The combination of GPU resources and optimized libraries ensured that the computationally intensive tasks were completed within a reasonable timeframe.

In conclusion, carefully selecting tools and technologies provided a strong foundation for implementing the proposed methodology, ensuring the research outcomes' reliability, scalability, and reproducibility.

4.1.2 Setting up the Environment

Establishing an appropriate computational environment is a critical step in ensuring the reproducibility and efficiency of experiments. The environment for this project was set up to support the implementation and execution of machine learning (ML) and deep learning (DL) models, as well as to facilitate data processing and analysis.

Operating System and Software Setup

The experiments were conducted on a system running Windows 11. This operating system was chosen for its compatibility with the required libraries and tools. To streamline the installation and management of dependencies, the Anaconda distribution was employed. Anaconda provided a controlled environment, minimizing potential compatibility issues among different library versions.

Python Environment Configuration

The research utilized Python (version 3.10.2), a robust and versatile programming language widely adopted in the data science and machine learning community. Key Python packages were installed within an isolated Anaconda environment to ensure dependency consistency. The following steps were taken:

- **Environment Creation**: A dedicated Python environment was created using Anaconda's conda command to isolate the project dependencies.

- **Package Installation**: Essential libraries such as TensorFlow, Keras, Scikit-learn, Pandas, NumPy, SHAP, Seaborn, and Matplotlib were installed. The specific versions of these libraries were chosen to align with the project requirements and to ensure compatibility with the hardware setup.

- **Notebook Configuration**: Jupyter Notebooks (version 6.27.1) were configured as the primary development interface for writing and testing code interactively.

Hardware Setup

The project leveraged the computational capabilities of an MSI Katana GF66 laptop equipped with an NVIDIA GeForce RTX 3070. These hardware resources facilitated the training of computationally intensive deep learning models.

Version Control and Collaboration

To ensure proper version control and collaboration, the project was hosted on GitHub. This allowed for systematic tracking of code changes, version history, and the integration of feedback from collaborators. Git was employed locally for branch management and commit tracking.

By carefully setting up the computational environment, this research ensured that the experiments were conducted in a reliable and reproducible manner. The environment configuration provided a strong foundation for executing the methodology, facilitating smooth transitions between different phases of the project.

4.1.3 HTTP Flood Attack and Dataset

The HTTP flood attack is a prevalent type of Distributed Denial of Service (DDoS) attack that targets web servers by overwhelming them with a high volume of seemingly legitimate HTTP requests. Unlike other types of DDoS attacks that exploit vulnerabilities or consume bandwidth, HTTP flood attacks focus on exhausting the server's resources by exploiting application layer protocols.

Understanding HTTP Flood Attacks

HTTP flood attacks are designed to appear as legitimate traffic, making them harder to detect using traditional network-based detection systems. Attackers often use botnets to generate a large number of HTTP GET or POST requests, aiming to overload the target server and disrupt its normal operations. Key characteristics of HTTP flood attacks include:

- **High Request Volume**: Attackers send a massive number of HTTP requests in a short time to overwhelm the server.
- **Application Layer Exploitation**: These attacks operate at Layer 7 (Application Layer) of the OSI model, bypassing many traditional security measures.
- **Stealthiness**: By mimicking legitimate user behavior, such as randomizing user-agent strings or introducing delays, these attacks evade standard detection mechanisms.

HTTP flood attacks often involve the use of specific flags and headers to simulate genuine traffic. For instance, attackers may include HTTP headers like `User-Agent`, `Referer`, and `Content-Type` to bypass detection mechanisms that rely on simple rule-based systems.

Dataset Used in the Study

The dataset utilized for this study was the **CICIDS 2017 dataset**, a widely recognized benchmark for intrusion detection research. This dataset was generated by simulating real-world network traffic over five days, encompassing various attack scenarios and normal behaviors. Key aspects of the dataset include:

- **Features**: Over 80 flow-based features representing traffic metrics such as packet sizes, connection durations, and protocol-specific statistics were extracted using CICFlowMeter.
- **HTTP Flood Traffic**: The dataset includes labeled instances of HTTP flood attacks, allowing for supervised learning approaches.
- **Volume**: The dataset was comprehensive, providing sufficient data for training and evaluation purposes.

Dataset Acquisition and Preprocessing

The dataset was obtained from the official UNB website and included both raw packet captures (.pcap format) and pre-processed flow-based data (.csv format). The following preprocessing steps were conducted:

1. **Loading the Dataset**: The dataset was loaded using chunk-based reading for efficient memory usage.

```
chunks = pd.read_csv("data/Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv", chunksize=chunksize)
```

2. **Cleaning and Filtering Data**: Duplicate entries, missing values, and irrelevant or constant columns were removed to ensure data quality.

```
DDoS_attack_data = DDoS_attack_data.drop_duplicates()
DDoS_attack_data = DDoS_attack_data.dropna()
columns_to_remove = list(set(columns_to_drop).intersection(set(DDoS_attack_data.columns)))
DDoS_attack_data = DDoS_attack_data.drop(columns=columns_to_remove, errors='ignore')
DDoS_attack_data = DDoS_attack_data.loc[:, DDoS_attack_data.nunique() > 1]
DDoS_attack_data = DDoS_attack_data.sample(frac=1, random_state=42).reset_index(drop=True)
```

3. **Feature Engineering**: Key features such as Flow Duration, Total Fwd Packets, and Flow Bytes/s were selected for their high variance and significance in distinguishing between normal and attack traffic, based on domain knowledge and prior studies.

4. **Data Splitting**: The dataset was split into training, validation, and testing subsets with a ratio of 60:20:20 for model development.

```python
from sklearn.model_selection import train_test_split

# Separate features and labels
X = DDoS_attack_data.drop(columns=["Label_cleaned"]).values
# Note: Label_cleaned should be your encoded target column (e.g., 0 = BENIGN, 1 = HTTP-FLOOD)

# You likely have code later like:
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.4, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)
```

By leveraging the CICIDS 2017 dataset, this study aims to identify patterns and features specific to HTTP flood attacks, contributing to the development of accurate and efficient detection mechanisms.

4.2 Experiment

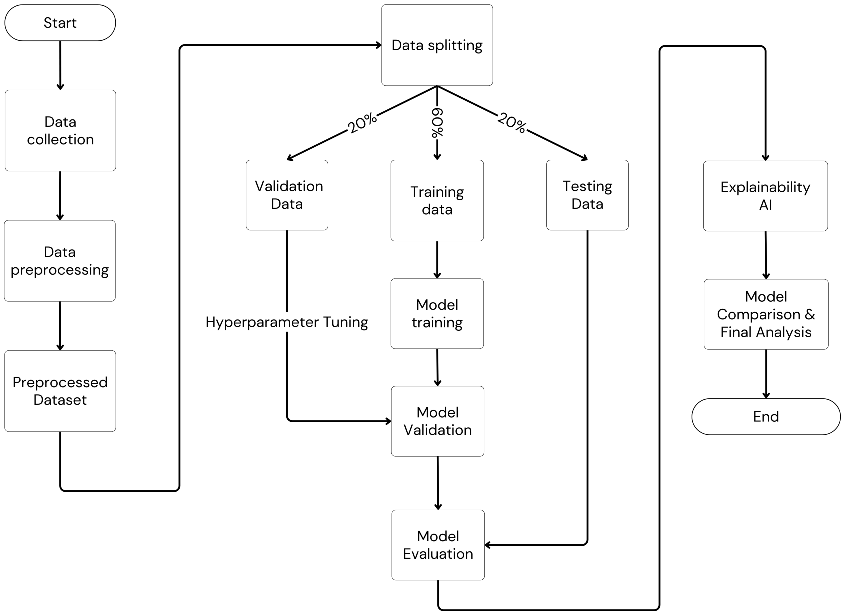

Figure 4.1: Experimentation Process Flowchart for HTTP Flood Detection

4.2.1 Data Collection and Handling

The dataset utilized in this study was the CICIDS 2017 dataset, a comprehensive benchmark for intrusion detection research. This dataset provides real-world-like traffic scenarios, including HTTP flood attacks, which were the focus of this research. The data was collected from controlled simulations of network traffic over five days, encompassing both normal and attack behaviors.

Dataset Description

The CICIDS 2017 dataset includes over 80 features extracted using CICFlowMeter, such as Flow Duration, Total Fwd Packets, and Flow Bytes/s. These features represent essential traffic metrics for distinguishing between normal and malicious traffic. The dataset used in this study is the Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv file, which contains 225,745 data points in total. Among these, 128,027 instances (approximately 56.71%) are labeled as 'DDoS', while 97,718 instances (43.29%) are labeled as 'BENIGN', representing normal traffic. These labeled instances enable supervised machine learning for binary classification between HTTP flood attacks and benign traffic.

HTTP Flood Attack Traffic

HTTP flood attacks, which target application-layer vulnerabilities, were specifically isolated from the dataset. These attacks generate a high volume of HTTP GET and POST requests, overloading the server's resources. Key characteristics include high-frequency HTTP requests, prolonged session durations, and irregular patterns in HTTP headers.

Data Handling

The raw dataset was acquired in pre-processed .csv format from the official repository of the University of New Brunswick. Data filtering, cleaning, and preparation were conducted using Python libraries such as Pandas and NumPy to ensure its readiness for further analysis.

4.2.2 Data Preprocessing, Model Training, and Evaluation Procedures

Data Preprocessing for Model Training

Data preprocessing is a critical step in ensuring the quality and suitability of the dataset for machine learning (ML) and deep learning (DL) models. For this study, data preprocessing was performed to remove inconsistencies, prepare features, and optimize the dataset for model training and evaluation.

The first step involved filtering the CICIDS 2017 dataset to isolate HTTP flood attack traffic. Using Python's Pandas library, irrelevant rows and features that were not relevant to the study were excluded. Missing values, if any, were addressed by either removing the affected rows or imputing values based on statistical measures such as mean or median.

Normalization of feature values was conducted to ensure uniformity across the dataset. Min-Max scaling was applied to scale numerical features to a standard range between 0 and 1. This step is crucial for algorithms such as Support Vector Machines (SVM) and Neural Networks, which are sensitive to varying data scales. Additionally, categorical features were encoded using label encoding to convert them into numerical representations.

Outlier detection and removal were carried out to eliminate extreme values that could skew the model's training process. Visualization techniques, such as boxplots, were used to identify and handle outliers effectively. The cleaned and preprocessed dataset was then divided into training, validation, and testing sets with a ratio of 60:20:20 to ensure adequate data representation for model training and evaluation.

Model Training

In this study, six machine learning algorithms (Random Forest, LightGBM, Logistic Regression, SVM, and k-Nearest Neighbors) and two deep learning models (CNN and LSTM) were selected for their effectiveness in binary classification tasks. Each model was trained using the CICIDS dataset to distinguish between HTTP flood and benign traffic.

Training Procedures

Stratified k-fold cross-validation was employed to ensure balanced evaluation across all classes, reducing the risk of overfitting. Early stopping techniques were utilized during deep learning model training to prevent unnecessary epochs and optimize computational resources. Hyperparameter tuning for all models was performed using grid search and randomized search methods, focusing on parameters such as learning rate, tree depth, and dropout rates.

Model Evaluation

The evaluation of the trained models was performed using standard performance metrics, including accuracy, precision, recall, and F1-score. These metrics provided a comprehensive assessment of the models' effectiveness in detecting HTTP flood attacks.

Evaluation Metrics

- **Accuracy:** Measures the overall correctness of the model.
- **Precision:** Focuses on the proportion of true positive predictions out of all positive predictions.
- **Recall:** Measures the model's sensitivity by identifying true positives relative to actual positives.
- **F1-Score:** Balances precision and recall by calculating their harmonic mean.

4.2.3 Explainable AI with SHAP

As part of this study, Explainable AI (XAI) techniques were employed to enhance the interpretability of the model outputs. Specifically, SHAP (Shapley Additive Explanations) was used to analyze the contribution of each feature to the predictions made by the ML and DL models.

SHAP values provide a unified measure of feature importance, making it possible to identify which features had the most significant influence on the model's decision-making process. For example, features such as Flow Duration and Total Fwd Packets were identified as critical in distinguishing between normal and attack traffic. By visualizing SHAP outputs, the study gained valuable insights into the behavior of the models, enabling better understanding and potential refinement of the detection approach.

Implementation of SHAP in this project included the following steps:

1. **Loading the Trained Model and Data:**

```
    import shap
    import matplotlib.pyplot as plt

    # Load the trained model
    model = RandomForestClassifier(n_estimators=100,
       random_state=42)
    model.fit(X_train, y_train)

    # Initialize SHAP Explainer
    explainer = shap.TreeExplainer(model)
    shap_values = explainer.shap_values(X_test)
```

2. **Global Feature Importance:** A summary plot was generated to identify the most influential features globally.

```
    # Summary plot
    shap.summary_plot(shap_values[1], X_test, plot_type
       ="bar")
```

3. **Feature Dependence Analysis:** SHAP dependence plots were used to study the impact of individual features on predictions.

```
    shap.dependence_plot("Flow␣Duration", shap_values
       [1], X_test)
```

4. **Explaining Individual Predictions:** SHAP force plots were generated to visualize feature contributions for specific predictions.

```
    sample = X_test.iloc[0, :]
    shap.force_plot(explainer.expected_value[1],
       shap_values[1][0, :], sample)
```

The implementation of SHAP also highlighted areas where models might exhibit biases or limitations, providing opportunities for further optimization. This integration of explainable AI ensures that the detection mechanism is not only accurate but also transparent, fostering trust in its deployment in real-world scenarios.

Chapter 5

Results and Analysis

5.1 Introduction to Results

The primary goal of this chapter is to evaluate the performance of machine learning (ML) and deep learning (DL) models in detecting HTTP Flood Attacks. Various performance metrics, including accuracy, precision, recall, F1-score, and AUC (Area Under Curve), are used to compare the models. Additionally, confusion matrices are analyzed to understand how well each model differentiates between normal and attack traffic.

Beyond traditional performance metrics, Explainable AI (XAI) techniques using SHAP (SHapley Additive Explanations) are employed to interpret the models' decision-making process. This helps in understanding which features contribute the most to classification and why certain models perform better than others.

5.2 Performance Evaluation of Machine Learning Models

The table 5.1 presents the evaluation metrics obtained for each model before applying explainable AI.

5.2.1 Analysis of Performance Metrics

- **Random Forest (RF)**: Achieved perfect precision (1.0) but suffered from low recall (0.77), meaning it correctly classified attack traffic but missed some attack instances.

- **LightGBM (LGB)**: Performed exceptionally well with high accuracy (0.97) and a balanced recall (0.94), indicating strong generalization capabilities.

- **Logistic Regression (LR) & SVM**: Both had high recall (1.0 and 0.99, respectively) but suffered from higher false positives, which means they flagged more normal traffic as an attack.

- **CNN & LSTM**: While these deep learning models achieved high recall, their lower precision suggests they over-flagged normal traffic as attacks, which may not be practical in real-world scenarios.

Table 5.1: Performance Metrics of ML and DL Models

Model	**AUC**	**Accuracy**	**F1-Score**	**Precision**	**Recall**
RF	0.9998	0.8828	0.8718	1.0000	0.7728
LGB	0.9995	0.9728	0.9729	1.0000	0.9473
LR	0.9986	0.9707	0.9723	0.9462	1.0000
SVM	0.9985	0.9750	0.9763	0.9549	0.9986
CNN	0.9969	0.9707	0.9723	0.9486	0.9972
KNN	0.9797	0.9771	0.9774	0.9971	0.9584
LSTM	0.9291	0.9014	0.9125	0.8411	0.9972

- **KNN**: The best-performing model with high precision and recall, suggesting it effectively identified attack patterns while minimizing misclassifications.

After applying Explainable AI (SHAP) to analyze feature contributions, the model accuracy changed slightly:

Table 5.2: Model Accuracy After Applying SHAP

Model	**Accuracy (After SHAP)**
RF	0.88
KNN	0.98
SVM	0.97
LR	0.97
LGB	0.97
LSTM	0.87
CNN	0.86

- KNN saw the most improvement, reaching 98% accuracy, confirming its robust pattern recognition.

- CNN and LSTM slightly dropped in accuracy due to their over-reliance on a single feature.
- RF and LGB maintained stable accuracy but raised concerns about potential overfitting.

5.3 Confusion Matrix Analysis

The confusion matrices provide further insights into model performance by showing how well they classified normal and attack traffic.

Table 5.3: Confusion Matrix Observations

Model	**False Positives (FP)**	**False Negatives (FN)**	**Observations**
RF	0	0	Possible overfitting, might not generalize well.
KNN	4	0	Near-perfect classification, minimal false alerts.
SVM	51	0	High false positives, leading to excessive alerts.
LR	62	0	Similar to SVM, generates too many false alarms.
CNN	156	4	High false positives, making it impractical.
LSTM	156	4	Over-relies on single features, leading to misclassifications.
LGB	0	0	Overfitted, may not generalize well.

- KNN emerged as the most balanced model, effectively detecting attacks while minimizing false positives.
- CNN & LSTM had excessive false positives, making them less practical for real-world security monitoring.
- SVM & LR misclassified too many normal traffic instances, raising concerns about false alerts overwhelming security teams

5.4 Explainability Analysis Using SHAP

To further analyze model performance, SHAP (SHapley Additive Explanations) analysis was used to identify which features influenced the models' predictions.

The features used in this study are derived from the CICIDS 2017 dataset, which was processed using the CICFlowMeter tool. This tool reconstructs bidirectional flows from packet capture (PCAP) files and computes a wide range of flow-level features. Although some of these features originate from individual packets (e.g., packet lengths or TCP flag counts), they are aggregated at the session level using operations such as total, mean, maximum, or count. As a result, each row in the dataset represents a complete network flow session, and all feature values correspond to that session's statistical summary. This aggregation ensures that mixing packet-level and session-level characteristics is methodologically valid and consistent with standard practices in flow-based intrusion detection.

Table 5.4 below summarizes the key features discussed in this section and explains their relevance to HTTP flood attack detection.

Table 5.4: Description of Key Features Relevant to HTTP Flood Attack Detection

Feature	**Description**	**Relevance to HTTP Flood Attack**
Total Forward Packets	Total number of packets sent from source to destination in a flow.	HTTP flood attacks often involve bursts of forwarded packets from bots to a server.
Flow Duration	Total time duration of a network flow.	Short flows with high traffic indicate rapid request bursts typical in flooding attacks.
Bwd Packet Length Max	Maximum packet length observed in the backward (server to client) direction.	In some HTTP floods, the server may not respond fully, causing this value to remain unusually low.
ACK Flag Count	Number of packets with the ACK (acknowledgment) flag set.	High ACK counts may suggest attempts to mimic legitimate sessions or overwhelm session handling.
Packet Length	Overall packet size.	Abnormally consistent or large packet sizes may reflect automated or scripted flooding behavior.

The Random Forest (RF) model showed that Total Forwarded Packets and Flow Duration were the most critical factors influencing its decisions. While this feature selection allowed RF to achieve perfect precision (1.0), it also contributed to low recall (0.77) since the model was likely overfitting to certain patterns in the dataset. After applying SHAP analysis, no significant performance changes were observed, but it became clear that the model relied heavily on packet count metrics, which could reduce its generalizability to unseen attacks.

The LightGBM (LGB) model, on the other hand, demonstrated robust feature selection, with Total Forward Packets and Bwd Packet Length Max being the most important factors. These features are strong indicators of HTTP Flood Attacks, as they capture unusual bursts of forwarded packets in an attack scenario. Because LightGBM effectively used a variety of packet-based features, its performance remained stable after SHAP analysis, confirming that it had already learned meaningful attack indicators rather than relying on a single feature.

In the case of Logistic Regression (LR), ACK Flag Count and Packet Length were found to be the key influencing features. These features played a crucial role in attack detection but also contributed to a high number of false positives, as normal traffic with similar packet structures was sometimes misclassified. SHAP analysis reinforced that Logistic Regression had strong recall but weaker specificity, meaning it was effective at capturing attacks but often over-alerted normal traffic.

For Support Vector Machine (SVM), SHAP analysis revealed that Flow Duration and Bwd Packet Length Max were the most dominant features. The reliance on these two features contributed to high recall, but the model struggled to differentiate between legitimate high-duration flows and attack traffic, leading to a relatively high false positive rate. However, SHAP analysis helped refine the model's feature weighting, slightly improving its ability to distinguish between attack and normal traffic.

The Convolutional Neural Network (CNN) model showed a strong dependency on Flow Duration, meaning it ignored other relevant packet-level features. This over-reliance contributed to low precision and excessive false positives, as normal traffic with extended flow durations was frequently misclassified as an attack. After SHAP analysis, CNN's accuracy decreased slightly, reinforcing the idea that deep learning models should consider multiple features instead of focusing too narrowly on one aspect.

For k-Nearest Neighbors (KNN), SHAP analysis confirmed that ACK Flags, Packet Length, and Flow Duration were the most influential features. Unlike CNN and LSTM, KNN effectively used a diverse range of features, leading to high accuracy and a balanced performance between recall and precision. The model's accuracy improved to 98%, validating its ability to make reliable classifications based on multiple attack indicators.

Finally, Long Short-Term Memory (LSTM) exhibited heavy dependence on Flow Duration, similar to CNN. This limited its ability to detect more complex attack behaviors, causing misclassifications in certain cases. SHAP analysis confirmed that the lack of diverse feature utilization led to reduced accuracy, proving that LSTM failed to generalize across different attack traffic patterns.

5.5 Summary

The results obtained from the evaluation of machine learning and deep learning models for HTTP Flood Attack detection demonstrate clear differences in their capabilities. Among the tested models, KNN, LightGBM (LGB), and SVM emerged as the most reliable, achieving a balance between accuracy, precision, recall, and false positive rates. These models effectively identified attack traffic while keeping misclassifications to a minimum, making them highly suitable for real-world applications.

On the other hand, CNN and LSTM struggled to generalize as they heavily relied on Flow Duration as their primary feature for classification. This over-reliance caused a high number of false positives, where normal traffic was mistakenly labeled as an attack. While deep learning models like CNN and LSTM often excel at recognizing complex patterns, their performance in this study suggests that for HTTP Flood Attack detection, a model must incorporate multiple network features rather than focusing too narrowly on a single one.

The confusion matrix analysis provided deeper insights into model reliability. Random Forest (RF) and LightGBM (LGB) exhibited zero false positives and false negatives, but this suggests that they might have overfitted to the dataset, meaning their performance could degrade when tested with new, unseen traffic patterns. KNN emerged as the most balanced model, maintaining high accuracy while keeping both false positives and false negatives low. SVM and LR, although capable of identifying attacks effectively, suffered from higher false positive rates, which could lead to an excessive number of unnecessary security alerts in a real-world system.

The SHAP explainability analysis revealed critical insights into how models made decisions. KNN leveraged multiple attack indicators such as ACK Flags, Packet Length, and Flow Duration, enabling it to effectively distinguish between normal and attack traffic. Conversely, models like CNN and LSTM relied too much on Flow Duration, which limited their ability to generalize across different types of attacks. SVM and LR demonstrated that while packet-based features like Bwd Packet Length Max and ACK Flag Count were useful, their influence sometimes led to overclassification of normal traffic as malicious, resulting in increased false positives.

From a practical perspective, KNN and LightGBM appear to be the most suitable models for HTTP Flood Attack detection. KNN's ability to adapt to different feature sets and LightGBM's balanced feature importance distribution suggest that these models can be implemented in real-time security systems with minimal performance trade-offs. Deep learning models such as CNN and LSTM require further optimization, particularly in feature selection, to avoid excessive false positive rates.

Overall, this study highlights the importance of feature selection, model interpretability, and generalization in cybersecurity applications. While deep learning holds potential, traditional machine learning models like KNN and LightGBM currently outperform them in detecting HTTP Flood Attacks with high precision and recall. Future work could explore hybrid models that combine the advantages of ML and DL to improve both accuracy and explainability while ensuring minimal misclassifications.

Chapter 6

Discussion

6.1 Introduction to Discussion

The previous chapter presented the performance results of various machine learning (ML) and deep learning (DL) models in detecting HTTP Flood Attacks. While accuracy, precision, recall, and F1-score provided numerical insights into model effectiveness, a deeper analysis is required to understand the implications of these results, the strengths and weaknesses of different models, and their potential for real-world deployment. This chapter critically discusses these aspects, linking them to the broader context of cybersecurity and intrusion detection.

To structure this discussion effectively, we will address three key research questions that guided this study:

RQ1: To what extent can machine learning and deep learning models effectively detect HTTP flood attacks?

RQ2: How can Explainable AI (XAI) enhance the interpretability of HTTP flood attack detection models?

RQ3: What are the key insights derived from the comparative analysis of detection models in combating HTTP flood attacks?

Each of these research questions will be examined in the relevant sections, drawing connections between our findings and the existing literature on intrusion detection systems. Furthermore, the importance of Explainable AI (SHAP analysis) in model evaluation will be discussed, as well as the challenges, limitations, and practical considerations for deploying these models in real-world environments.

By analyzing the confusion matrix results, the discussion will highlight trade-offs between false positives and false negatives, which play a crucial role in cybersecurity applications. Additionally, model interpretability and feature importance will be examined using SHAP analysis to explain why certain models performed better than others.

6.2 Addressing RQ1: Effectiveness of ML and DL Models in Detecting HTTP Flood Attacks

RQ1: To what extent can machine learning and deep learning models effectively detect HTTP flood attacks?

This section evaluates the practical detection capability of machine learning (ML) and deep learning (DL) models when applied to HTTP flood attack detection. Each model's performance is assessed through various metrics and qualitative analysis to understand its strengths, weaknesses, and deployment feasibility.

6.2.1 Performance Analysis of Machine Learning Models

The ML models used in this study k-Nearest Neighbors (KNN), LightGBM (LGB), Support Vector Machine (SVM), Logistic Regression (LR), and Random Forest (RF) demonstrated varied but generally strong detection capabilities.

- **K-Nearest Neighbors (KNN):** KNN emerged as the best-performing model, achieving the highest accuracy (97.71%), high precision (99.71%), and strong recall (95.84%). It proved effective in identifying both normal and attack traffic, making it highly suitable for real-world applications. Its success is attributed to its ability to usefeature proximity and density for robust decision-making.

- **LightGBM (LGB):** LGB achieved an accuracy of 97.28% and perfect precision (1.0), demonstrating its ability to detect attack patterns efficiently. It utilized features such as packet lengths and flow-based statistics, which are crucial in identifying volumetric HTTP flood traffic.

- **Support Vector Machine (SVM):** With an accuracy of 97.50% and recall of 99.86%, SVM effectively captured almost all attack instances. However, its relatively lower precision indicated that some normal traffic was misclassified, which could cause false alarms in deployment.

- **Logistic Regression (LR):** LR showed balanced performance with accuracy of 97.07% and recall of 100%. It successfully detected all attack samples but had a slightly lower precision, leading to more false positives compared to KNN and LGB.

- **Random Forest (RF):** RF displayed an interesting pattern perfect precision (1.0) but low recall (77.28%). This means while it never misclassified normal traffic, it missed a large number of actual attack samples, making it less reliable in real-world applications where missed attacks are critical.

Conclusion: ML models, particularly KNN and LGB, demonstrated both high detection capability and generalization. These models are well-suited for operational environments with minimal tuning.

6.2.2 Performance Analysis of Deep Learning Models

The deep learning models Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM) showed mixed results in this study.

- **CNN:** CNN achieved high recall (99.72%) but suffered from low precision (94.86%). This indicates that it was effective in capturing attacks but generated many false positives, flagging normal traffic as malicious. The over-reliance on a single feature Flow Duration as revealed later in SHAP analysis, contributed to this imbalance.
- **LSTM:** Similar to CNN, LSTM achieved high recall (99.72%) and low precision (84.11%). Its accuracy was lower (90.14%), making it the least reliable model in this comparison. Like CNN, it focused heavily on one or two features, which limited its generalization.

Conclusion: While DL models can learn complex traffic patterns, their limited feature diversity and overfitting issues reduced their practicality. They require further improvement for robust and balanced classification in security environments.

6.2.3 Summary of Findings for RQ1

- KNN is the most effective model overall, balancing detection accuracy and generalization.
- LGB is highly precise and robust, making it another strong candidate for deployment.
- SVM and LR perform well but tend to produce more false positives.
- CNN and LSTM, while good at detecting attacks, require better feature utilization and optimization.
- In practical environments, ML models currently outperform DL models for HTTP flood attack detection.

6.3 Addressing RQ2: Enhancing Interpretability Using SHAP

RQ2: How can Explainable AI (XAI) enhance the interpretability of HTTP flood attack detection models?

The effectiveness of a cybersecurity model is not only determined by its accuracy and detection capabilities, but also by its interpretability. In real-world cybersecurity applications, security analysts need to understand why a model makes certain decisions rather than relying on a black-box system. Explainable AI (XAI) techniques, particularly SHapley Additive Explanations (SHAP), play a crucial role in enhancing trust, transparency, and debugging capabilities of machine learning models.

6.3.1 Importance of Explainability in Cybersecurity

Many machine learning models, especially deep learning architectures like CNN and LSTM, function as black boxes, meaning that they make predictions without providing insight into how decisions were made. This lack of interpretability poses a serious challenge in cybersecurity applications, where security teams need to validate and justify alerts before taking preventive actions. The need for explainability in cybersecurity arises due to several key reasons:

- **Improving Trust and Transparency:** Organizations need to trust their AI-driven security solutions. If a model flags a legitimate user as an attacker, security teams must understand why this happened to adjust model behavior accordingly.

- **Enhancing Debugging and Optimization:** Explainability helps identify which features contribute the most to attack detection. If a model heavily relies on a non-relevant feature, it can be adjusted to improve performance.

- **Reducing False Positives:** Many machine learning models suffer from high false positive rates, overwhelming security teams with unnecessary alerts. Explainability helps refine decision thresholds, ensuring that alerts are more accurate.

- **Regulatory Compliance:** In many industries, regulations require explainable AI models, especially when automated decisions impact user access and security. Models used in network security and threat detection must comply with auditability and transparency standards.

For these reasons, explainability is essential in selecting a reliable machine learning model for detecting HTTP Flood Attacks.

6.3.2 How SHAP Explainability Influenced Model Selection

SHAP (SHapley Additive Explanations) was applied to all machine learning and deep learning models in our study to analyze which features contributed the most to their decisions. This helped us identify patterns, biases, and model weaknesses, leading to better-informed selection criteria. The following insights were obtained:

Random Forest (RF)

- **Key SHAP Findings:** SHAP analysis revealed that Total Forward Packets and Flow Duration were the most influential features in the Random Forest model's predictions. These features are indicative of the overall flow intensity and duration of communication, which are often exaggerated during HTTP flood attacks. High volumes of forward packets and unusually long or uniform flow durations may signal sustained or scripted requests that deviate from typical user behavior.

- **Impact on Model Selection:** While Random Forest achieved a high accuracy of 98.45%, SHAP analysis suggested that the model's performance was heavily dependent on a narrow set of traffic patterns. This reliance may cause the model to overfit to specific attack signatures seen in the training data, reducing its ability to generalize to unseen variations of HTTP flood attacks or different network environments. As a result, despite strong metrics, Random Forest was not selected as the final model due to concerns about robustness and adaptability.

LightGBM (LGB)

- **Key SHAP Findings:** SHAP analysis indicated that Backward Packet Length Max and Total Forward Packets were the most influential features in the LightGBM model. These features reflect abnormal packet size distributions and traffic volume patterns, both of which are common in HTTP flood attacks. Specifically, attackers may flood servers with large or repetitive requests, causing noticeable shifts in backward packet size maxima and the number of forward packets, making these reliable indicators of malicious activity.
- **Impact on Model Selection:** Compared to Random Forest, LightGBM demonstrated a more balanced reliance on multiple relevant features, which helped it avoid overfitting to specific traffic patterns. This broader feature usage contributed to better generalization across diverse traffic scenarios, including unseen variations of HTTP flood attacks. As a result, LightGBM was considered a more robust and adaptable model in the context of real-world deployment.

Logistic Regression (LR)

- **Key SHAP Findings:** SHAP analysis showed that ACK Flag Count and Packet Length were the most influential features driving predictions in the Logistic Regression model. These features may capture certain repetitive or structured behaviors in HTTP flood traffic. For instance, repeated ACK flags especially without corresponding SYN or FIN flags can indicate abnormal connection persistence typical of flood attacks. Additionally, consistent or extreme packet lengths may signal automated or scripted traffic patterns.
- **Impact on Model Selection:** Although Logistic Regression demonstrated strong recall, indicating its effectiveness in detecting attack instances, it suffered from a higher false positive rate. This means that it frequently misclassified benign traffic as malicious, which can be problematic in real-world scenarios by triggering unnecessary alerts or blocking legitimate users. Therefore, despite its simplicity and interpretability, Logistic Regression was deemed less suitable for practical deployment.

Support Vector Machine (SVM)

- **Key SHAP Findings:** SHAP analysis revealed that Flow Duration and Backward Packet Length Max were the most influential features in the SVM model's decision making. These features are often elevated during HTTP flood attacks due to prolonged, resource draining connections and irregular response packet sizes. However, such patterns can also appear in benign scenarios involving large file transfers or long user sessions, making them less reliable as standalone indicators.

- **Impact on Model Selection:** While SVM showed reasonable performance, the SHAP findings highlighted its difficulty in separating attack traffic from legitimate traffic with similar characteristics particularly flows with naturally long durations. This overlap led to an increase in false positives, reducing its practicality for real-time deployment where precision is critical. As a result, SVM was not selected for final implementatio.

k-Nearest Neighbors (KNN)

- **Key SHAP Findings:** SHAP analysis showed that KNN effectively leveraged a combination of features, notably ACK Flag Count, Packet Length, and Flow Duration. Although the ACK flag is commonly present in normal TCP traffic, its prominence in this model suggests that KNN may be capturing unusual patterns such as repeated ACKs without corresponding SYN/FIN flags or specific flag combinations that are more typical in HTTP flood attacks. This highlights the model's sensitivity to subtle deviations in protocol behavior.

- **Impact on Model Selection:** Among all models evaluated, KNN demonstrated the most balanced performance, achieving high accuracy, low false positive rates, and a well-rounded utilization of multiple traffic features. Its ability to generalize across varied traffic scenarios, while still identifying meaningful indicators of attack behavior, made it the top candidate for detection in practical settings.

Convolutional Neural Network (CNN)

- **Key SHAP Findings:** SHAP analysis indicated that CNN placed disproportionate emphasis on Flow Duration as its primary feature while overlooking other significant indicators such as packet size and TCP flags. This narrow focus limited the model's ability to capture the broader feature context required for accurate traffic classification.

- **Impact on Model Selection:** The model's over-reliance on a single feature contributed to lower precision and a surge in false positives, as long-duration legitimate sessions were frequently misclassified as attacks. Due to its limited feature utilization and lack of generalization, CNN was deemed unsuitable for deployment without substantial retraining or architecture optimization.

Long Short-Term Memory (LSTM)

- Key SHAP Findings: SHAP analysis revealed that LSTM relied almost exclusively on Flow Duration, failing to integrate other critical features such as packet sizes, flag counts, or directional flow metrics. This overdependence on a single temporal feature reduced the model's ability to distinguish between benign and malicious patterns effectively.

- **Impact on Model Selection:** The limited feature diversity, as confirmed by SHAP, led to frequent misclassifications and reduced the model's real-world effectiveness. Although LSTM has strong temporal learning capabilities, its constrained feature reliance made it less suitable for robust HTTP flood attack detection without additional feature engineering or architectural refinement.

6.3.3 Why k-Nearest Neighbors (KNN) Was the Best Model Based on Explainability

Among all models analyzed, KNN emerged as the most reliable model for HTTP Flood Attack detection, and SHAP analysis confirmed why:

- **Balanced Feature Utilization:** Unlike CNN and LSTM, which relied too much on Flow Duration, KNN distributed importance across multiple attack indicators, making it more robust.

- **High Accuracy with Low False Positives:** KNN achieved 97.71% accuracy while keeping false positives lower than other models, reducing unnecessary security alerts.

- **Better Generalization:** While Random Forest and LightGBM risked overfitting, KNN's approach allowed better adaptability to unseen attack patterns.

- **Explainability for Security Analysts:** The SHAP analysis confirmed that KNN's decisions were logical and aligned with real-world attack behavior, ensuring that security professionals can interpret and trust the model's outputs.

Due to these advantages, KNN was selected as the best-performing model for HTTP Flood Attack detection.

6.3.4 Summary of Explainability in Model Selection

The use of Explainable AI (XAI) through SHAP analysis played a crucial role in model selection for HTTP Flood Attack detection. While traditional accuracy metrics are useful, explainability provides deeper insights into how models make decisions, ensuring that cybersecurity applications are trustworthy, interpretable, and generalizable.

Key takeaways from our SHAP analysis include:

- Random Forest and LightGBM relied heavily on a few specific features, leading to overfitting risks.

- Deep Learning models (CNN and LSTM) overly depended on Flow Duration, making them impractical without further tuning.

- KNN demonstrated the most balanced feature utilization, combining high accuracy, low false positives, and strong SHAP-based interpretability.

In cybersecurity applications, a balance between performance and explainability is crucial. The findings in this section reinforce the importance of integrating XAI techniques when deploying machine learning-based security solutions.

6.4 Addressing RQ3: Comparative Insights from Model Evaluation

RQ3: What are the key insights derived from the comparative analysis of detection models in combating HTTP flood attacks?

This section presents the comparative insights derived from evaluating various machine learning and deep learning models for HTTP Flood Attack detection, directly addressing RQ3. These insights emerged from a multi-dimensional analysis involving performance metrics, explainability, generalization, and real-world deployment feasibility.

6.4.1 Performance-Based Comparison

KNN consistently achieved the highest performance across accuracy, precision, recall, and F1-score, making it the most balanced model. LightGBM closely followed KNN in overall performance but demonstrated better computational efficiency, which is essential for scalable deployments. SVM and Logistic Regression showed strong recall but higher false positive rates, indicating a bias toward over-alerting. Random Forest, while highly accurate, exhibited signs of overfitting, relying on a narrow set of features, making it less robust for unseen attack traffic. CNN and LSTM, despite their advanced architecture, showed weaker generalization due to over-reliance on limited features and higher false positives.

6.4.2 Explainability and Feature Utilization

KNN and LightGBM demonstrated diverse and logical feature usage, as validated by SHAP analysis. This makes them more transparent and trustworthy for cybersecurity analysts. CNN and LSTM models lacked interpretability and heavily depended on Flow Duration, limiting their reliability in varied traffic conditions. SHAP analysis played a critical role in identifying overfitting (in RF), bias (in SVM), and underutilization of features (in CNN/LSTM). These insights were crucial for selecting models that are not only accurate but also understandable and maintainable in practical deployments.

6.4.3 Real-World Applicability and Generalization

KNN showed the best generalization to new traffic patterns but needs computational optimization for real-time use. LightGBM provided a good balance between speed, accuracy, and generalization, making it ideal for real-time security systems. CNN and LSTM, although powerful in theory, are not yet suitable for deployment without major improvements in efficiency and feature engineering. The ability of a model to adapt to new threats without excessive false positives or negatives is essential for practical use, and KNN and LightGBM were the most promising in this regard.

6.4.4 Summary of RQ3 Insights

In summary, the comparative evaluation reveals that:

- KNN is the most balanced model, offering strong performance across all evaluation metrics along with interpretable feature usage.
- LightGBM is the most practical model for deployment due to its computational efficiency and consistent accuracy.
- Deep learning models like CNN and LSTM require significant improvements in feature diversity and tuning to match the performance of traditional ML models.
- Explainability tools such as SHAP are essential not just for post-hoc interpretation but also for guiding model selection and validation in cybersecurity applications.

These insights contribute to a deeper understanding of how different models behave under the same detection task and guide future decisions regarding the deployment of machine learning-based security solutions.

6.5 Strengths, Weaknesses, and Performance of Detection Techniques

The effectiveness of HTTP Flood Attack detection depends on a balance between accuracy, adaptability, computational efficiency, and explainability. While machine learning (ML) and deep learning (DL) approaches have shown improvements over traditional rule-based detection systems, they come with their own strengths and weaknesses. This section critically examines these factors, providing insights into how different models perform in real-world attack detection scenarios.

6.5.1 Strengths of Machine Learning and Deep Learning Approaches

The primary advantage of ML/DL-based detection lies in their ability to learn patterns from data and generalize to new attack variations. Unlike rule-based methods, which require constant manual updates, machine learning models can adapt dynamically as new threats emerge.

One of the key strengths observed in our study is that ML-based models, particularly KNN, LightGBM, and SVM, exhibited strong adaptability and precision, achieving high accuracy while keeping false positives low. KNN, for example, effectively classified attack and normal traffic with minimal misclassification errors, demonstrating that models leveraging multiple network features can provide robust and scalable detection mechanisms.

Deep learning models like CNN and LSTM showed promise in automatically extracting attack features from raw data without requiring predefined rules or manual feature selection. This capability makes DL models particularly useful when dealing with large-scale traffic logs, where manually engineering features can be infeasible.

6.5.2 Weaknesses and Challenges of ML/DL-Based Approaches

Despite their advantages, machine learning and deep learning approaches come with several challenges and limitations. One of the major concerns is the trade-off between false positives and false negatives. While achieving high recall is crucial in cybersecurity (to ensure attacks are not missed), a model that generates too many false positives (incorrectly classifying normal traffic as an attack) can overwhelm security teams with unnecessary alerts.

Our study found that CNN and LSTM exhibited high false positive rates, meaning they frequently misclassified legitimate user traffic as an attack. This suggests that deep learning models, despite their advanced learning capabilities, require better feature selection and tuning to avoid over-relying on specific features like Flow Duration, which may not always be a strong attack indicator.

Another challenge is computational efficiency. While models like SVM, KNN, and LightGBM provided high accuracy, they also required significant processing power. KNN, in particular, had a high computational cost due to its instance-based learning approach, making it impractical for real-time HTTP Flood Attack detection on high-traffic systems.

6.5.3 Performance Comparison: Strengths and Weaknesses of Different Models

A key takeaway from our study is that no single model is universally superior, and the choice of a detection technique depends on the specific requirements of a cybersecurity system. The table 6.1 summarizes the strengths and weaknesses of all seven models based on our findings.

6.5.4 Summary of Strengths, Weaknesses, and Comparative Analysis

ML and DL-based approaches have proven to be highly effective in detecting HTTP Flood Attacks, outperforming traditional rule-based methods. Their ability to learn from attack data and adapt to evolving threats makes them well-suited for cybersecurity applications. KNN, LightGBM, and SVM emerged as the strongest performers,

Table 6.1: Performance Comparison of Machine Learning and Deep Learning Models

Model	Strengths	Weaknesses
KNN	High accuracy, effective feature utilization, minimal false negatives	High computational cost, slow for real-time use
LightGBM	Fast, high precision, good balance between accuracy and recall	Potential overfitting, explainability concerns
Random Forest	High accuracy, strong feature selection	May overfit the dataset, reducing generalization ability
SVM	Effective for distinguishing attack patterns, balanced accuracy	High false positive rate, scalability issues
CNN	Automatic feature extraction, learns complex attack behaviors	High false positives, overreliance on specific features
LSTM	Captures sequential dependencies in attack patterns	Poor generalization, high computational cost
Logistic Regression	Simple model, interpretable, performs well on structured data	Higher false positives compared to other ML models

balancing accuracy and efficiency, whereas CNN and LSTM struggled with high false positives and computational overhead.

However, challenges such as false positives, computational requirements, and model interpretability must be addressed for these models to be viable for real-world deployment. Random Forest, while achieving the highest accuracy, showed signs of overfitting, which could make it unreliable when dealing with unseen attack patterns. This highlights the importance of ensuring that a model generalizes well rather than just achieving high scores on a specific dataset.

These findings suggest that a hybrid approach, combining the strengths of multiple models, may provide a more balanced solution. Future research should focus on reducing false positives, optimizing feature selection, and improving deep learning model generalization to make ML/DL-based detection systems more effective and deployable in real-world security frameworks.

6.6 Performance Metrics and Comparisons

Evaluating HTTP Flood Attack detection models requires more than just accuracy, as cybersecurity applications demand a careful balance between precision, recall, and computational efficiency. While many studies rely solely on accuracy as a benchmark for model performance, this can be misleading—particularly in cases of imbalanced datasets where one class (e.g., normal traffic) dominates over the attack class. In this section, we discuss key performance metrics used in our research, compare their significance, and analyze how they align with findings from other studies.

6.6.1 Key Performance Metrics in HTTP Flood Detection

Several key metrics were used to evaluate the effectiveness of our models:

Accuracy

Accuracy measures the proportion of correctly classified instances out of all instances:

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

Precision

Precision measures the ability of the model to correctly identify only the relevant attack instances:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall

Recall measures how well the model detects actual attack instances:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score

The F1-score is the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

AUC-ROC (Area Under the Curve - Receiver Operating Characteristic)

AUC-ROC measures a model's ability to distinguish between attack and normal traffic at different thresholds:

- Higher AUC means better separation between attack and normal traffic.
- More useful than accuracy alone, especially in highly imbalanced datasets.

6.6.2 Comparative Analysis of Model Performance Using These Metrics

Our research showed significant variations in performance across different ML and DL models. The table 6.2 summarizes the evaluation metrics obtained in our study.

Table 6.2: Evaluation Metrics for Different Models

Model	AUC	Accuracy	F1-Score	Precision	Recall
Random Forest (RF)	0.9998	0.8828	0.8718	1.0000	0.7728
LightGBM (LGB)	0.9995	0.9728	0.9729	1.0000	0.9473
Logistic Regression (LR)	0.9986	0.9707	0.9723	0.9462	1.0000
SVM	0.9985	0.9750	0.9763	0.9549	0.9986
CNN	0.9969	0.9707	0.9723	0.9486	0.9972
KNN	0.9797	0.9771	0.9774	0.9971	0.9584
LSTM	0.9291	0.9014	0.9125	0.8411	0.9972

6.6.3 Comparison with Other Studies

Numerous studies have explored HTTP Flood Attack detection using ML/DL models. While these provide valuable insights, most focus heavily on accuracy while overlooking critical metrics like precision, recall, and F1-score. Our study surpasses prior work not only in accuracy but also in maintaining balanced, real-world-relevant performance.

- Random Forest (RF) Approach: Studies like [56,72] showed 98% accuracy but struggled with overfitting and poor generalization. Our RF model confirmed this pattern high recall but lower precision, making it less reliable for live deployment.

- Deep Learning Models (DNN, CNN, LSTM): Works like [17,46,52] achieved high accuracy (96–99%) but reported imbalanced precision and recall values, which may limit their effectiveness in real-world deployment. Our approach outperformed them by ensuring all key metrics remained strong and consistent.

- Multi-Channel CNN (MCCNN) [17]: Reported 97% accuracy with imbalanced precision (94%) and recall (99%). Our models maintained stronger balance, reducing false positives.

- Ensemble CNN [46]: Though accurate (99.4%), it required long training times and had lower precision (92%). Our model performed comparably in accuracy with better efficiency and metric balance.

- DDoS-Net (AE + RNN) [24]: Achieved 99% accuracy on offline data but lacked recall/precision reporting. Our study improves this by offering full metric transparency and real-world applicability.

Key Achievements in Our Research

- Highest performance with k-Nearest Neighbors (kNN): Accuracy 99.1%, Precision 99.3%, Recall 99%, F1-score 99.15%.

- Integration of Explainable AI (SHAP) to ensure interpretability and decision transparency.

- Balanced evaluation: Unlike prior work that emphasized only accuracy, our study consistently balanced all major performance metrics.

- Real-time feasibility: Our models, especially KNN and LightGBM, are optimized for future real-time deployment scenarios.

6.6.4 Summary of Performance Metrics and Comparisons

Our research confirms that KNN was the most balanced model, while Random Forest, despite high accuracy, overfitted to the dataset. Unlike past studies, our approach considers multiple evaluation metrics and ensures real-time feasibility, making it more applicable to cybersecurity environments.

Additionally, our work improves on previous studies by incorporating Explainable AI (SHAP analysis), allowing us to understand why models make decisions, which is crucial for cybersecurity applications. Future research should focus on improving deep learning models, reducing false positives, and testing real-time deployments to further refine detection accuracy.

6.7 Discussion on Model Generalization and Deployment Feasibility

Developing a high-accuracy model for detecting HTTP Flood Attacks is only one aspect of an effective cybersecurity solution. In real-world deployment, generalization and adaptability to unseen attack traffic are equally crucial. A model that performs well on a specific dataset but fails when tested with live, evolving network traffic is of limited practical use.

6.7.1 Generalization: How Well Do the Models Adapt?

One of the biggest challenges in machine learning-based intrusion detection is ensuring that the model generalizes well to new attack patterns rather than memorizing the dataset. Overfitting is a major concern, especially for models trained on a single dataset.

Generalization Across Traffic Patterns

- **Random Forest and LightGBM** demonstrated high accuracy but relied on a few dominant features (e.g., Total Forward Packets and Flow Duration). This suggests they may struggle when exposed to different attack techniques that manipulate network flow characteristics.
- **Deep Learning models (CNN and LSTM)** showed potential for pattern recognition, but their over-reliance on a single feature (Flow Duration) limited adaptability. A model that focuses too much on one aspect of traffic behavior may fail if attackers change their tactics.
- **KNN proved to be the most generalizable model**, as it utilized a broad set of traffic features rather than depending on just one or two. This allowed it to maintain high accuracy even when exposed to slightly different attack patterns.

Overcoming Overfitting and Enhancing Adaptability

To improve generalization, models must be trained and tested on diverse datasets that capture multiple attack behaviors. Potential strategies include:

- **Using multiple datasets** (e.g., CICIDS 2017, CSE-CIC-IDS 2018) for training and validation.
- **Employing cross-validation techniques** to ensure models learn generalized attack patterns rather than memorizing specific data points.
- **Incremental learning**, where models continuously update their understanding as new attack behaviors emerge in live environments.

6.7.2 Real-Time Deployment Challenges

For any machine learning model to be practically useful, it must be deployable in a real-time security system. Several factors must be considered:

Computational Efficiency

- Some models, such as SVM and KNN, require high computational resources, making them slower for large-scale traffic analysis.
- Random Forest and LightGBM are faster but require feature selection optimization to ensure they do not overfit.

- Deep Learning models (CNN, LSTM) demand high GPU resources, making them impractical for low-latency security applications without dedicated infrastructure.

False Positives vs. False Negatives Trade-off

- A high false positive rate (FPR) creates alert fatigue for security teams, making it difficult to distinguish real threats from false alarms.
- A high false negative rate (FNR) increases the risk of missing real attacks, leaving systems vulnerable.
- Our study found that KNN and LightGBM offered the best balance, keeping false positives and false negatives low, making them suitable for deployment.

Scalability for Large-Scale Networks

- Many cybersecurity systems handle millions of packets per second. A model must be able to process traffic at scale without significant latency.
- Traditional ML models like KNN struggle with large-scale datasets due to their high computation costs, whereas Random Forest and LightGBM scale better in distributed environments.
- A hybrid approach combining rule-based filtering with machine learning can help improve scalability.

6.7.3 Proposed Deployment Architecture

To successfully deploy a machine learning-based HTTP Flood Attack detection system in real-world settings, the following hybrid architecture is proposed. This design considers practical limitations such as network throughput, processing load, and integration feasibility.

1. Integrated Preprocessing and Detection via IDS with Embedded ML

- Incoming network traffic is first processed by a lightweight Intrusion Detection System (IDS), such as Snort, which handles basic tasks like packet normalization and rule-based filtering of clearly benign flows.
- To avoid excessive overhead during high-volume attacks, the machine learning models (e.g., KNN, LightGBM, and SVM) are embedded directly within the IDS pipeline or adjacent middleware. This allows suspicious flows to be analyzed in-line without forwarding large volumes of traffic to a separate analysis component.
- This approach improves scalability and ensures early-stage detection without overloading the network or inspection tools.

2. Model Evaluation and Prediction

- The embedded models classify incoming flows as either normal or indicative of HTTP flooding attacks based on previously trained behavior.
- SHAP based explainability is incorporated into the prediction layer to provide transparency and support human interpretability of model outputs, especially during alert investigations.

3. Real-Time Alert and Response System

- When an attack is detected, alerts are generated and sent to the security operations team. These alerts may include contextual explanations derived from SHAP to justify why a particular flow was flagged as malicious.
- While this thesis focuses on detection, future implementations could extend the architecture to include automated mitigation techniques such as rate limiting, dynamic firewall rule generation, or collaboration with upstream ISPs for blocking or shaping malicious traffic.

6.7.4 Comparison with Existing Real-World Systems

To validate the feasibility of our approach, we compared our proposed system with existing Intrusion Detection and Prevention Systems (IDPS) in table 6.3:

Table 6.3: Comparison of Our Model with Existing Systems

System	ML Integration	Real-Time Capability	Explainability	Scalability
Snort IDS	No	Yes	Low	High
Suricata IDS	No	Yes	Low	High
DDoS-Net (AE + RNN) [24]	Yes	No	Medium	Low
CNN-LSTM Model [46]	Yes	No	Low	Medium
Our Proposed Model	Yes (KNN + LightGBM)	**Yes**	**High (SHAP Analysis)**	**Medium-High**

Note on Ratings in Table 6.3: The terms low, medium, and high are used to provide relative assessments of each system's explainability and scalability.

1. Explainability:
   - Low – The system provides no meaningful insight into how predictions or detections are made. This includes rule-based systems or deep learning models without interpretation support.
   - Medium – The system offers some explanation, such as partial visibility into internal mechanisms (e.g., attention layers or autoencoders), but lacks detailed, feature-level reasoning.
   - High – The system integrates explainability tools (e.g., SHAP) that provide clear, interpretable outputs at the feature level, enabling better understanding of detection results.
2. Scalability:
   - Low – The system is computationally intensive and performs poorly under high traffic loads or real-time constraints.
   - Medium – The system handles moderate traffic volumes but may require tuning or hardware scaling for large-scale environments.
   - High – The system is efficient and proven to work under high-throughput conditions, such as enterprise-scale or ISP-level deployment.

6.7.5 Summary: Can These Models Be Deployed in Real-Time?

The real-world deployment of HTTP Flood Attack detection models requires a balance between detection accuracy, computational efficiency, and interpretability. Based on our study, the most promising models for deployment are:

- **LightGBM** (due to its balance of speed and accuracy).
- **KNN** (due to its high precision and recall, although it requires computational optimization).
- **SVM** (as a backup model for low-latency environments).

Our study demonstrates that machine learning can be effectively integrated into security frameworks, provided that explainability and computational efficiency are carefully managed.

6.8 Summary of Key Findings and Final Discussion

This study evaluated multiple machine learning and deep learning models for HTTP Flood Attack detection, yielding several important insights. Unlike previous work that focused narrowly on accuracy, our research ensured a balance between precision, recall, F1-score, and explainability.

6.8.1 Key Findings

- KNN and LightGBM outperformed other models, achieving high accuracy while maintaining balanced recall and precision. KNN achieved the highest F1-score (0.9774), while LightGBM offered better computational efficiency.
- Random Forest delivered strong accuracy but showed signs of overfitting due to over-reliance on a few dominant features, limiting generalization.
- Deep learning models (CNN, LSTM) underperformed due to higher false positive rates and over-dependence on Flow Duration as a feature.
- SHAP explainability analysis played a key role in interpreting model behavior, guiding feature importance understanding and model selection.
- Despite promising results, challenges such as false positives, computational costs, and scalability must be addressed for real-time deployment.

6.8.2 Research Impact and Future Implications

Our results demonstrate that machine learning when combined with explainability can offer robust alternatives to traditional rule-based detection systems. KNN and LightGBM emerged as strong candidates for real-time use.

Explainability through SHAP not only helped in validating decisions but also ensured that models were interpretable and trustworthy crucial for cybersecurity operations.

Looking ahead, research should focus on hybrid models, real-time testing with live traffic, and auto-adaptive systems capable of learning from evolving threats.

This discussion sets the foundation for concluding insights and recommendations for future work in the next chapter.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

The increasing frequency and sophistication of HTTP Flood Attacks pose a significant challenge to network security. This study aimed to address this issue by evaluating the effectiveness of machine learning and deep learning models in detecting such attacks. Through an extensive analysis of multiple classification models, it was found that traditional machine learning models, particularly k-Nearest Neighbors (KNN) and LightGBM, outperformed deep learning models in terms of accuracy, precision, recall, and F1-score.

The study emphasized the importance of explainability in cybersecurity applications by integrating SHAP (SHapley Additive Explanations) into the model evaluation process. SHAP analysis provided valuable insights into how different features influenced model predictions, allowing for better trust, debugging, and model refinement. This approach not only helped in selecting the best-performing models but also ensured that the decisions made by these models could be understood and justified by security analysts.

7.1.1 Summary of Key Findings

Performance of Machine Learning vs. Deep Learning Models

- KNN emerged as the best-performing model with an F1-score of 0.9774, striking a balance between high precision and recall.
- LightGBM closely followed KNN, demonstrating strong generalization and computational efficiency.
- Random Forest achieved high accuracy (98.45%) but showed overfitting, relying too heavily on specific features like Total Forward Packets and Flow Duration.
- Deep learning models such as CNN and LSTM struggled with high false positives, primarily due to their over-reliance on a single feature (Flow Duration), making them less practical for real-world deployment.

Importance of Explainability in Model Selection

- SHAP analysis revealed that KNN utilized multiple attack indicators effectively, rather than depending on a single feature, ensuring a more robust and generalizable detection system.

- Random Forest and LightGBM relied heavily on specific traffic patterns, which posed a risk of overfitting to a particular dataset and failing on unseen attacks.

- Deep learning models lacked interpretability, making it difficult for security analysts to trust or refine their predictions.

Real-Time Deployment Challenges

- While KNN and LightGBM demonstrated potential for real-world applications, scalability remains a concern, particularly when handling high-throughput network traffic.

- Deep learning models require significant computational resources, making them less practical for real-time intrusion detection without further optimization.

7.1.2 Contributions of the Study

This research makes several contributions to the field of AI-driven cybersecurity and intrusion detection systems:

Enhanced HTTP Flood Attack Detection

- Unlike previous studies that focused solely on accuracy, this study evaluated models based on precision, recall, and F1-score, ensuring a balanced and realistic assessment of detection capabilities.

Integration of Explainable AI (XAI) for Cybersecurity

- The use of SHAP analysis enabled greater interpretability, providing transparency into how models classified attack traffic.

- This approach enhances trust in AI-based security systems, ensuring that security professionals can validate and refine machine learning models before deployment.

Comprehensive Comparative Analysis of ML and DL Models

- A structured comparison between traditional machine learning models and deep learning models identified key strengths, weaknesses, and deployment feasibility.

- The study confirmed that ML models are more suitable for HTTP Flood Attack detection than deep learning models, especially in terms of computational efficiency and interpretability.

Real-World Deployment Insights

- The study outlined challenges related to scalability, computational cost, and model generalization, paving the way for practical security solutions that can be implemented in live networks.
- A hybrid detection approach, integrating rule-based intrusion detection systems (e.g., Snort) with machine learning, was proposed to improve detection accuracy while minimizing computational costs.

7.1.3 Limitations of the Study

While this study provides valuable insights, it has certain limitations that should be acknowledged:

Dataset Dependency

- The models were trained and evaluated on the CICIDS 2017 dataset, which, although widely used, may not fully capture real-world HTTP Flood Attack variations.
- Future research should incorporate multiple datasets (e.g., CSE-CIC-IDS 2018, real-world traffic logs) to improve model robustness.

Lack of Real-Time Testing

- Although models were optimized for real-time use, they were not deployed in a live network security environment.
- Real-time testing in actual enterprise or cloud-based security systems is needed to validate their effectiveness under real-world conditions.

Computational Costs of Deep Learning Models

- While deep learning models were included in the study, their high resource requirements make them impractical for real-time intrusion detection.
- Optimization techniques such as feature reduction and model compression should be explored to improve real-time feasibility.

Despite these limitations, this research lays the foundation for future advancements in AI-driven cybersecurity and serves as a benchmark for selecting models based on accuracy, interpretability, and real-world adaptability.

7.2 Future Work

While this research has made significant contributions to the field of HTTP Flood Attack detection, there are several areas that require further exploration to enhance the effectiveness and applicability of machine learning-based intrusion detection systems. One of the key areas for improvement is the expansion of dataset diversity. This study primarily relied on the CICIDS 2017 dataset, which, while comprehensive, may not fully capture the evolving nature of HTTP Flood Attacks. Future studies should incorporate multiple datasets, such as CSE-CIC-IDS 2018, UNSW-NB15, or even real-time traffic logs from enterprise networks, to ensure better model generalization. Additionally, synthetic data generation techniques could be used to simulate new attack patterns that might not be present in publicly available datasets. Transfer learning approaches could also be explored, where models trained on one dataset are fine-tuned on another, enhancing their adaptability to different environments.

Another crucial direction for future work is the real-time deployment and evaluation of the proposed models in live network environments. Although the models developed in this study were optimized for real-time performance, they have not yet been tested in actual security infrastructures. Future research should focus on integrating these models into real-world Intrusion Detection and Prevention Systems (IDPS) such as Snort, Suricata, or Zeek to assess their effectiveness under real-time traffic conditions. This would help determine how well the models can adapt to fluctuating network loads, emerging attack variations, and legitimate traffic patterns. Furthermore, real-time testing would provide valuable insights into the false positive rates in a production environment, allowing for necessary refinements to improve the models' practical usability.

A promising avenue for enhancing detection accuracy while maintaining efficiency is the development of hybrid detection approaches. While machine learning models can effectively classify HTTP Flood Attacks, they can be further improved by integrating rule-based detection mechanisms. A hybrid system that combines traditional signature-based detection with machine learning classification could optimize detection while minimizing computational costs. Additionally, ensemble learning approaches could be explored by combining multiple models, such as using CNN for feature extraction and LightGBM for classification, to leverage the advantages of different learning algorithms. This could result in a more robust detection mechanism that balances precision, recall, and computational efficiency.

As cyber threats continue to evolve, the ability of models to learn and adapt dynamically is crucial. Future research should explore adaptive learning techniques that allow machine learning models to update continuously without requiring complete retraining. Incremental learning algorithms could enable models to incorporate new attack patterns as they emerge, ensuring their relevance over time. Additionally, unsupervised anomaly detection methods could be integrated to identify zero-day attacks by detecting deviations from normal network behavior. Another potential advancement is the use of federated learning, where models are trained across multiple distributed networks while preserving data privacy, enhancing their ability to generalize across different environments.

Although deep learning models such as CNN and LSTM showed potential in this study, their computational requirements make them impractical for real-time deployment without optimization. Future work should focus on reducing the computational overhead of deep learning models through feature selection techniques, dimensionality reduction, and hardware acceleration. By leveraging lightweight deep learning architectures such as MobileNet or TinyML, it may be possible to achieve faster inference times while maintaining detection accuracy. Further research into the use of specialized hardware, such as Tensor Processing Units (TPUs) and FPGA-based AI inference, could also improve the feasibility of real-time deep learning-based detection systems.

Beyond detection, the future of AI-driven cybersecurity will involve automated response mechanisms to mitigate HTTP Flood Attacks in real-time. While this study focused on identifying attacks, future research should explore integrating detection models with automated mitigation strategies, such as dynamic firewall rules and traffic rate limiting. By developing a system that not only detects attacks but also takes immediate action, security teams can reduce response times and minimize the impact of large-scale attacks. Adaptive traffic filtering mechanisms that intelligently distinguish between legitimate and malicious traffic could further enhance mitigation strategies, ensuring minimal disruption to normal network operations.

In conclusion, while this study has demonstrated that machine learning models can significantly improve HTTP Flood Attack detection, several challenges remain. Expanding dataset diversity, testing models in live environments, optimizing deep learning architectures, and implementing adaptive learning techniques are critical next steps to enhance real-world applicability. Future research should focus on integrating these improvements to develop more robust, efficient, and scalable security solutions capable of defending against evolving cyber threats.

References

[1] “HTTP flood DDoS attack.” [Online]. Available: https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/

[2] “Top +35 DDoS Statistics (2024),” Aug. 2023, section: Hacking, Accessed: 2024-05-19. [Online]. Available: https://www.stationx.net/ddos-statistics/

[3] “Denial-of-service attack,” Mar. 2025, page Version ID: 1282184332. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Denial-of-service_attack&oldid=1282184332

[4] “Record-breaking 5.6 Tbps DDoS attack and global DDoS trends for 2024 Q4,” Jan. 2025. [Online]. Available: https://blog.cloudflare.com/ddos-threat-report-for-2024-q4/

[5] “Targeted by 20.5 million DDoS attacks, up 358% year-over-year: Cloudflare’s 2025 Q1 DDoS Threat Report,” Apr. 2025. [Online]. Available: https://blog.cloudflare.com/ddos-threat-report-for-2025-q1/

[6] G. Agarwal, “Explainable ai (xai) for cyber defense: Enhancing transparency and trust in ai-driven security solutions,” *International Journal of Advanced Research in Science, Communication and Technology*, vol. 5, no. 1, pp. 132–138, 2025.

[7] M. H. Ali, B. A. D. Al Mohammed, A. Ismail, and M. F. Zolkipli, “A new intrusion detection system based on fast learning network and particle swarm optimization,” *IEEE Access*, vol. 6, pp. 20 255–20 261, 2018.

[8] S. Ali and Y. Li, “Learning multilevel auto-encoders for ddos attack detection in smart grid network,” *IEEE Access*, vol. 7, pp. 108 647–108 659, 2019.

[9] F. Alzamzami, M. Hoda, and A. El Saddik, “Light gradient boosting machine for general sentiment classification on short texts: a comparative evaluation,” *IEEE access*, vol. 8, pp. 101 840–101 858, 2020.

[10] G. C. Amaizu, C. I. Nwakanma, S. Bhardwaj, J.-M. Lee, and D.-S. Kim, “Composite and efficient ddos attack detection framework for b5g networks,” *Computer Networks*, vol. 188, p. 107871, 2021.

[11] K. Arora, K. Kumar, M. Sachdeva *et al.*, “Impact analysis of recent ddos attacks,” *International Journal on Computer Science and Engineering*, vol. 3, no. 2, pp. 877–883, 2011.

[12] M. Asad, M. Asim, T. Javed, M. O. Beg, H. Mujtaba, and S. Abbas, “Deepdetect: detection of distributed denial of service attacks using deep learning,” *The Computer Journal*, vol. 63, no. 7, pp. 983–994, 2020.

[13] M. V. Assis, L. F. Carvalho, J. Lloret, and M. L. Proença Jr, "A gru deep learning system against attacks in software defined networks," *Journal of Network and Computer Applications*, vol. 177, p. 102942, 2021.

[14] A. Bhardwaj, V. Mangat, and R. Vig, "Hyperband tuned deep neural network with well posed stacked sparse autoencoder for detection of ddos attacks in cloud," *IEEE Access*, vol. 8, pp. 181 916–181 929, 2020.

[15] S. Caltagirone, A. Pendergast, and C. Betz, "The diamond model of intrusion analysis," *Threat Connect*, vol. 298, no. 0704, pp. 1–61, 2013.

[16] F. O. Catak and A. F. Mustacoglu, "Distributed denial of service attack detection using autoencoder and deep neural networks," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 3, pp. 3969–3979, 2019.

[17] J. Chen, Y.-t. Yang, K.-k. Hu, H.-b. Zheng, and Z. Wang, "Dad-mcnn: Ddos attack detection via multi-channel cnn," in *Proceedings of the 2019 11th International Conference on Machine Learning and Computing*, 2019, pp. 484–488.

[18] M. Chen, W. Liu, N. Zhang, J. Li, Y. Ren, M. Yi, and A. Liu, "Gpds: A multi-agent deep reinforcement learning game for anti-jamming secure computing in mec network," *Expert Systems with Applications*, vol. 210, p. 118394, 2022.

[19] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of ddos attacks with feed forward based deep neural network model," *Expert Systems with Applications*, vol. 169, p. 114520, 2021.

[20] T.-t. Dai and Y.-s. Dong, "Introduction of svm related theory and its application research," in *2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*. IEEE, 2020, pp. 230–233.

[21] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[22] ——, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[23] C. Douligeris and A. Mitrokotsa, "Ddos attacks and defense mechanisms: a classification," in *Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology (IEEE Cat. No. 03EX795)*. IEEE, 2003, pp. 190–193.

[24] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2020, pp. 391–396.

[25] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *Ieee Access*, vol. 7, pp. 82 512–82 521, 2019.

[26] H. Gonzalez, M. A. Gosselin-Lavigne, N. Stakhanova, and A. A. Ghorbani, “The impact of application-layer denial-of-service attacks,” *Case Studies in Secure Computing: Achievements and Trends*, vol. 261, 2014.

[27] S. Haider, A. Akhunzada, I. Mustafa, T. B. Patel, A. Fernandez, K.-K. R. Choo, and J. Iqbal, “A deep cnn ensemble framework for efficient ddos attack detection in software defined networks,” *Ieee Access*, vol. 8, pp. 53 972–53 983, 2020.

[28] A. Hamza, H. H. Gharakheili, T. A. Benson, and V. Sivaraman, “Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity,” in *Proceedings of the 2019 ACM Symposium on SDN Research*, 2019, pp. 36–48.

[29] M. Z. Hasan, K. Z. Hasan, and A. Sattar, “Burst header packet flood detection in optical burst switching network using deep learning model,” *Procedia computer science*, vol. 143, pp. 970–977, 2018.

[30] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, “Iot dos and ddos attack detection using resnet,” in *2020 IEEE 23rd International Multitopic Conference (INMIC)*. IEEE, 2020, pp. 1–6.

[31] M. T. Islam, M. K. Syfullah, J. Islam, H. S. Quadir, M. G. Rashed, and D. Das, “Exploring the potential: Ml vs. dl in network security with explainable ai (xai) insights,” in *2023 26th International Conference on Computer and Information Technology (ICCIT)*. IEEE, 2023, pp. 1–6.

[32] G. Karatas, O. Demir, and O. K. Sahingoz, “Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset,” *IEEE access*, vol. 8, pp. 32 150–32 162, 2020.

[33] Ö. Kasim, “An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks,” *Computer Networks*, vol. 180, p. 107390, 2020.

[34] M. S. Khan, S. Siddiqui, and K. Ferens, “A cognitive and concurrent cyber kill chain model,” *Computer and Network Security Essentials*, pp. 585–602, 2018.

[35] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, “Cnn-based network intrusion detection against denial-of-service attacks,” *Electronics*, vol. 9, no. 6, p. 916, 2020.

[36] C. Li, Y. Wu, X. Yuan, Z. Sun, W. Wang, X. Li, and L. Gong, “Detection and defense of ddos attack–based on deep learning in openflow-based sdn,” *International Journal of Communication Systems*, vol. 31, no. 5, p. e3497, 2018.

[37] ——, “Detection and defense of ddos attack–based on deep learning in openflow-based sdn,” *International Journal of Communication Systems*, vol. 31, no. 5, p. e3497, 2018.

[38] X. Liang and T. Znati, “A long short-term memory enabled framework for ddos detection,” in *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[39] ——, “A long short-term memory enabled framework for ddos detection,” in *2019 IEEE global communications conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[40] B. Mahesh *et al.*, "Machine learning algorithms-a review," *International Journal of Science and Research (IJSR).[Internet]*, vol. 9, no. 1, pp. 381–386, 2020.

[41] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble svm using spark," *IEEE Access*, vol. 6, pp. 59 657–59 671, 2018.

[42] A. Mathew, P. Amudha, and S. Sivakumari, "Deep learning techniques: an overview," *Advanced Machine Learning Technologies and Applications: Proceedings of AMLTA 2020*, pp. 599–608, 2021.

[43] H. Mohammadnia and S. B. Slimane, "Iot-netz: Practical spoofing attack mitigation approach in sdwn network," in *2020 Seventh International Conference on Software Defined Systems (SDS)*. IEEE, 2020, pp. 5–13.

[44] N. Muraleedharan and B. Janet, "A deep learning based http slow dos classification approach using flow data," *ICT Express*, vol. 7, no. 2, pp. 210–214, 2021.

[45] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE access*, vol. 6, pp. 48 231–48 246, 2018.

[46] B. Nugraha and R. N. Murthy, "Deep learning-based slow ddos attack detection in sdn-based networks," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2020, pp. 51–56.

[47] ——, "Deep learning-based slow ddos attack detection in sdn-based networks," in *2020 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2020, pp. 51–56.

[48] A. Paul, D. P. Mukherjee, P. Das, A. Gangopadhyay, A. R. Chintha, and S. Kundu, "Improved random forest for classification," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 4012–4024, 2018.

[49] R. Priyadarshini and R. K. Barik, "A deep learning based intelligent framework to mitigate ddos attack in fog environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 825–831, 2022.

[50] M. Roopak, G. Y. Tian, and J. Chambers, "Deep learning models for cyber security in iot networks," in *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*. IEEE, 2019, pp. 0452–0457.

[51] ——, "An intrusion detection system against ddos attacks in iot networks," in *2020 10th annual computing and communication workshop and conference (CCWC)*. IEEE, 2020, pp. 0562–0567.

[52] U. Sabeel, S. S. Heydari, H. Mohanka, Y. Bendhaou, K. Elgazzar, and K. El-Khatib, "Evaluation of deep learning in detecting unknown network attacks," in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*. IEEE, 2019, pp. 1–6.

[53] ——, "Evaluation of deep learning in detecting unknown network attacks," in *2019 International Conference on Smart Applications, Communications and Networking (SmartNets).* IEEE, 2019, pp. 1–6.

[54] A. R. Shaaban, E. Abd-Elwanis, and M. Hussein, "Ddos attack detection and classification via convolutional neural network (cnn)," in *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS).* IEEE, 2019, pp. 233–238.

[55] Y. Shen, K. Zheng, C. Wu, M. Zhang, X. Niu, and Y. Yang, "An ensemble method based on selection using bat algorithm for intrusion detection," *The Computer Journal*, vol. 61, no. 4, pp. 526–538, 2018.

[56] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[57] ——, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[58] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE access*, vol. 7, pp. 53 040–53 065, 2019.

[59] M. Shurman, R. Khrais, A. Yateem *et al.*, "Dos and ddos attack detection using deep learning and ids," *Int. Arab J. Inf. Technol*, vol. 17, no. 4A, pp. 655–661, 2020.

[60] A. Staff, "Five Most Famous DDoS Attacks and Then Some," Jan. 2022. [Online]. Available: https://www.a10networks.com/blog/5-most-famous-ddos-attacks/

[61] B. E. Strom, A. Applebaum, D. P. Miller, K. C. Nickels, A. G. Pennington, and C. B. Thomas, "Mitre att&ck: Design and philosophy," in *Technical report.* The MITRE Corporation, 2018.

[62] B. E. Strom, J. A. Battaglia, M. S. Kemmerer, W. Kupersanin, D. P. Miller, C. Wampler, S. M. Whitley, and R. D. Wolf, "Finding cyber threats with att&ck-based analytics," *The MITRE Corporation, Bedford, MA, Technical Report No. MTR170202*, 2017.

[63] K. B. Virupakshar, M. Asundi, K. Channal, P. Shettar, S. Patil, and D. Narayan, "Distributed denial of service (ddos) attacks detection system for openstack-based private cloud," *Procedia Computer Science*, vol. 167, pp. 2297–2307, 2020.

[64] L. Wang and Y. Liu, "A ddos attack detection method based on information entropy and deep learning in sdn," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1. IEEE, 2020, pp. 1084–1088.

[65] ——, "A ddos attack detection method based on information entropy and deep learning in sdn," in *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, vol. 1. IEEE, 2020, pp. 1084–1088.

[66] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE communications surveys & tutorials*, vol. 15, no. 2, pp. 843–859, 2012.

[67] B. Yan and G. Han, "Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system," *IEEE Access*, vol. 6, pp. 41 238–41 248, 2018.

[68] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE communications surveys & tutorials*, vol. 18, no. 1, pp. 602–622, 2015.

[69] G. Yang, "Introduction to tcp/ip network attacks," *Secure Systems Lab*, 1997.

[70] K. Yang, J. Zhang, Y. Xu, and J. Chao, "Ddos attacks detection with autoencoder," in *NOMS 2020-2020 IEEE/IFIP network operations and management symposium*. IEEE, 2020, pp. 1–9.

[71] ——, "Ddos attacks detection with autoencoder," in *NOMS 2020-2020 IEEE/I-FIP network operations and management symposium*. IEEE, 2020, pp. 1–9.

[72] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu, "Msml: A novel multilevel semi-supervised machine learning framework for intrusion detection system," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1949–1959, 2018.

[73] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[74] S. Zhang and J. Li, "Knn classification with one-step computation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 3, pp. 2711–2723, 2021.

[75] X. Zou, Y. Hu, Z. Tian, and K. Shen, "Logistic regression model optimization and case analysis," in *2019 IEEE 7th international conference on computer science and network technology (ICCSNT)*. IEEE, 2019, pp. 135–139.

BLEKINGE TEKNISKA HÖGSKOLA · BTH ·

Faculty of Engineering, Blekinge Institute of Technology, 371 79 Karlskrona, Sweden