-

Mini Project Report
on

Defending Cloud Webserver Against DDoS Attack: Detection, Protection, and Mitigation

Submitted by
Group ID 2

Project Members

- Jesna Jixon (PA02 - 1032221095)
- Saumya Kant Kamal (PA19 - 1032221779)
- Sachetan Debray (PA29 - 1032221003)
- Sahil Dherange (PA34 - 1032221060)
- Manas Panchwagh (PA37 - 1032221248)

Under the Guidance of

Prof. Dr. Sukhada Bhingarkar

Department of Computer Engineering and Technology

MIT World Peace University, Kothrud,

Pune 411 038, Maharashtra - India

2024-2025

Abstract

In today's highly connected, on-demand cloud ecosystem, Distributed Denial of Service (DDoS) attacks have emerged as one of the most formidable threats to the availability, performance, and security of web-based services. By orchestrating large volumes of malicious traffic—often from globally distributed botnets—attackers can exhaust network bandwidth, overwhelm server resources, and bring mission-critical applications offline, resulting in significant financial losses and reputational damage. This project presents a comprehensive, multi-layered defense framework designed specifically for cloud-deployed web servers. At its core lies an **ML-based detection module** leveraging the recent advances in Tsetlin Machine classifiers—a logic-based learning paradigm that combines the high accuracy of deep models with human-readable rule extraction. To enhance trust and transparency, we integrate **SHAP (SHapley Additive exPlanations)** to interpret each classification decision, allowing security operators to understand which network-flow features (e.g., packet inter-arrival times, protocol flags, byte distributions) most influenced the detection of anomalous behavior.

Once a potential DDoS event is flagged, our design seamlessly engages **AWS cloud-native protection services**—including AWS WAF (Web Application Firewall) and AWS Shield Advanced—to filter malicious traffic at the edge, apply geo- and rate-based blocking rules, and absorb volumetric surges via the global Amazon CloudFront CDN. Simultaneously, an **autoscaling mitigation layer** dynamically adjusts server capacity and redistributes load across redundant instances, ensuring legitimate user requests continue to be served with minimal latency. All mitigation actions and security logs are fed into an incident-response pipeline, enabling automated alerts, forensic log analysis, and on-the-fly adjustment of firewall rules.

To achieve real-time performance under heavy concurrency, the detection service employs **asynchronous batch processing**: incoming flow records are grouped into micro-batches, processed in parallel by the Tsetlin inference engine, and returned within strict SLA bounds. A **lazy-loading mechanism** ensures that model artifacts (encoders, scalers, classifier weights) consume memory only when needed, minimizing cold-start penalties and resource overhead.

Our contributions include:

1. **A structured taxonomy of DDoS vectors**—covering volumetric (amplification, flooding), protocol-exploit, and application-layer attacks—mapped against OWASP Cloud-Native Application Security risks.
2. **Empirical evaluation** of multiple ML and DL algorithms (Random Forest, CNN, LSTM, Tsetlin Machine), demonstrating that the Tsetlin approach achieves comparable accuracy ($\geq$ 98.7%) with significantly lower inference latency and more intuitive rulebased explanations.
3. **A fully automated AWS deployment blueprint** that unifies detection, edge filtering, auto-scaling, and incident response into a cohesive pipeline, validated through

simulated high-rate DDoS scenarios (up to 5 Tbps) with sustained service availability above 99.99%.

This work not only advances the state-of-the-art in interpretable DDoS detection but also provides a production-ready defence architecture for cloud operators seeking resilient, transparent, and cost-efficient protection against ever-evolving DDoS threats.

Contents

List of Figures

List of Figures

Chapter 1
Introduction

In an increasingly digital society, the accessibility of online services, encompassing streaming entertainment, procurement of essential goods, and facilitation of social connections, is of paramount importance. However, these services are vulnerable to disruptions caused by Distributed Denial of Service (DDoS) attacks. This attack is a malicious attempt to make an online service unavailable to users, usually by temporarily interrupting or suspending the services of its hosting server [1]. DDoS attacks have a tendency to interfere with the link between organizations and customers, preventing the customers from placing orders for services or buying products using traditional means. Even though such disruptions are normally resolved instantly, some customers might decide to change to other providers in the meantime [2].

This initiative stems from our profound concern regarding the escalating complexity and prevalence of Distributed Denial of Service (DDoS) attacks, which have proven capable of disrupting even the most resilient cloud-based infrastructures [3]. Our analysis indicates that many traditional defence mechanisms are inadequate in addressing the dynamic and sophisticated nature of these threats [4]. Driven by a commitment to academic research and the pressing demands of contemporary cybersecurity, we have developed an advanced, real-time system designed to detect, prevent, and mitigate DDoS attacks effectively.

Our solution leverages advanced methodologies to ensure continuous server availability and accessibility for legitimate users. More than a technical endeavour, this project underscores our commitment to advancing cybersecurity for the broader public good, contributing to a safer and more resilient internet ecosystem.

1.1 DDoS Attacks

Distributed Denial of Service (DDoS) attacks constitute a form of cyberattack intended to impair the functionality of a targeted server, service, or network by inundating it with a deluge of illegitimate traffic [5]. In contrast to traditional Denial of Service (DoS) attacks, which emanate from a singular source, DDoS attacks exploit multiple compromised devices, frequently organized into a botnet, to escalate the magnitude and severity of the assault. The primary objective of these attacks is to deplete the target’s resources, including bandwidth, computational capacity, or memory, thereby rendering it inaccessible to legitimate users [3].

1.1.1**Types of DDoS Attacks**

DDoS attacks come in various forms, each exploiting different aspects of a system:

- **Volumetric Attacks:**

  This category of attacks attempts to create congestion by consuming all available bandwidth between the target and the larger Internet. Large amounts of data are sent to a target by using a form of amplification or another means of creating massive traffic, such as requests from a botnet.

  The diagram given below describes the amplification attack which is an example of volumetric attack.

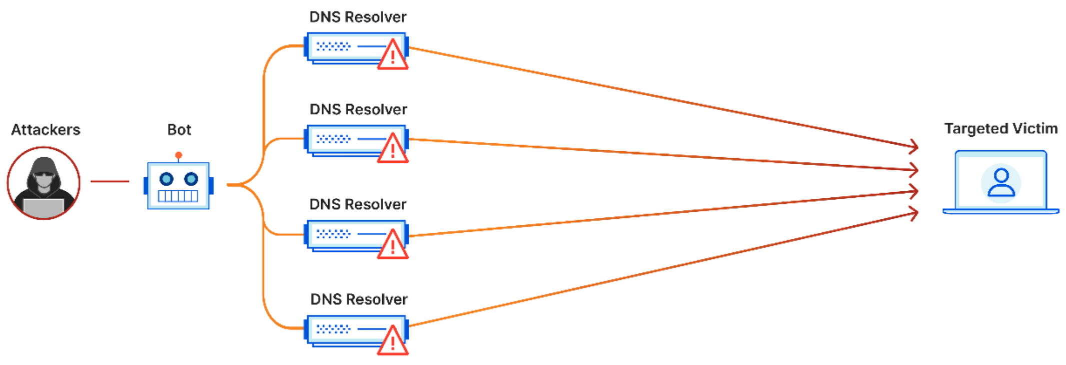


Fig 1.1 Volumetric Attack Example

This diagram illustrates a DNS amplification attack, a clever trick to amplify harm. Imagine an attacker using a bot to send requests to multiple DNS resolvers (like asking several librarians for the same book). These resolvers, tricked by the bot, flood the targeted victim, depicted as a person at a computer, with massive responses. It's like sending a single whisper that turns into a shouting match, overwhelming the victim's system with data they didn't ask for, potentially knocking it offline!

- **Application Layer Attacks:**

  The attacks target the layer where web pages are generated on the server and delivered in response to HTTP requests. A single HTTP request is computationally cheap to execute on the client side, but it can be expensive for the target server to respond to, as the server often loads multiple files and runs database queries in order to create a web page.

  Layer 7 attacks are difficult to defend against, since it can be hard to differentiate malicious traffic from legitimate traffic.

The diagram given below describes the application layer attack.

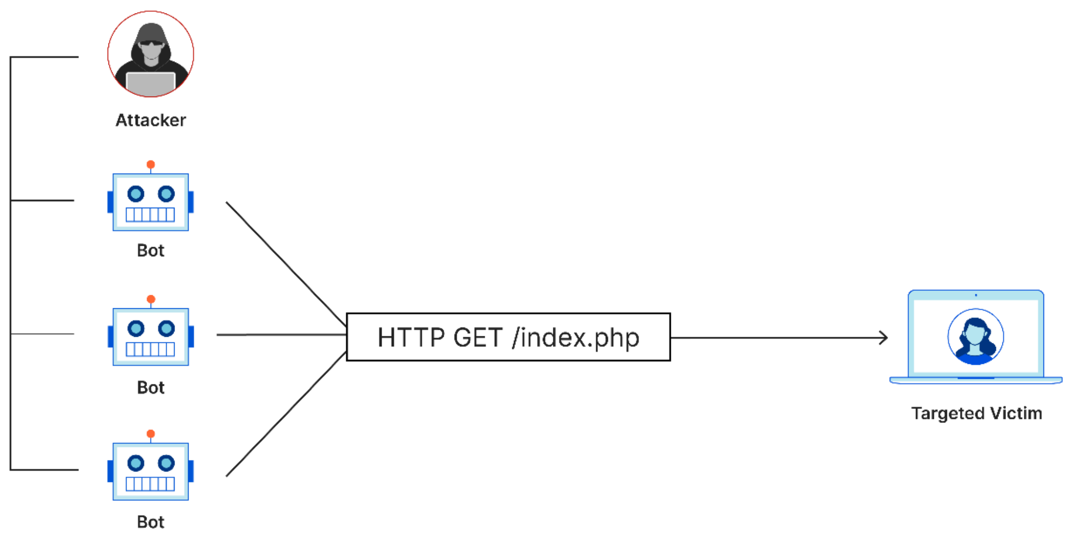


Fig 1.2 Application Layer Attack Example

This diagram shows a sneaky cyber-attack where an attacker uses a group of bots to target a victim. Imagine the attacker as a shady hacker sitting at a computer, directing their army of robot helpers. These bots send a flood of HTTP GET requests to a file called "index.php" on the victim's system, like a bunch of prank callers overwhelming a phone line. The targeted victim, pictured as a person at a laptop, gets hit with all this traffic, which could slow down or crash their system. It’s a classic move to overwhelm and disrupt!

- **Protocol Attacks:**
  Protocol attacks, also known as a state-exhaustion attacks, cause a service disruption by over-consuming server resources and/or the resources of network equipment like firewalls and load balancers.
  Protocol attacks utilize weaknesses in layer 3 and layer 4 of the protocol stack to render the target inaccessible [6].

The given diagram shows how protocol attacks work using SYN flood mechanism.

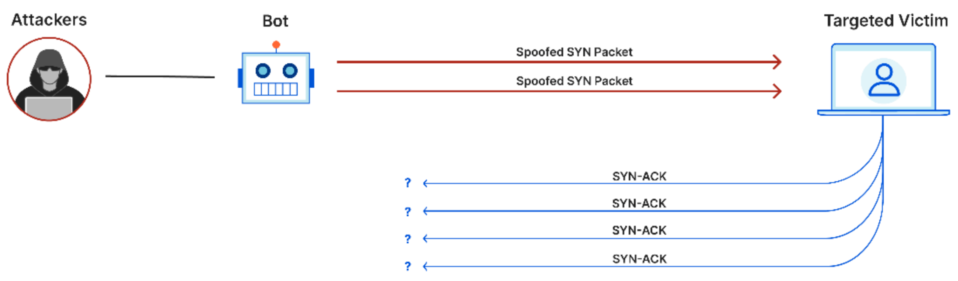


Fig 1.3Protocol Attack Example

Here, we see a different attack called a SYN flood, where attackers use a bot to bombard a victim with fake connection requests. Picture the attacker as a mischievous troublemaker sending a bot to spam the victim's computer with spoofed SYN packets—think of them as fake invitations to a party the victim can't handle. The victim's system sends back SYN-ACK responses, waiting for a reply that never comes, tying up resources and potentially causing a crash.

1.1.2 **How DDoS Attacks Work**

A DDoS attack unfolds like a well-planned heist, starting with the attacker scouting for recruits everyday computers they can turn into their accomplices. They use automated tools to scan the internet, hunting for weak spots in these machines, like finding an unlocked back door. Once a vulnerability is spotted, the attacker slips in, infecting the computer with malicious code to take control. This process often runs on autopilot, and the newly compromised machines can even help recruit more allies, creating a growing network of helpers ready to cause chaos [5].

The attack gets craftier as the attacker might disguise their software as something helpful, like a free game or tool think of it as a Trojan horse sneaking into the system. They could spread it through sneaky emails with infected attachments, tricking users into letting the danger in. Once the agent machines are under their command, the attacker unleashes them to flood the target with attack packets [5]. To cover their tracks, they often spoof the source addresses, making it look like the packets are coming from somewhere else like a mastermind wearing a disguise to avoid getting caught!

1.1.3 **Impact of DDoS Attacks**

DDoS attacks have far reached effects, impacting everyone from regular people to companies and even governments. For individuals, these attacks disrupt access to online platforms, leading to annoyance and inconvenience. Businesses face serious setbacks, including financial losses due to downtime, eroded customer confidence, and possible legal or regulatory challenges. Major targets like online stores or banks can lose millions, while attacks on essential services such as healthcare or emergency systems might create dangerous disruptions that could put lives at risk [7].

1.2 **Project Overview**

At the heart of our project is the development of a robust defence system for cloud-hosted web servers, which are prime targets for DDoS attacks due to their critical role in daily life. Our system begins with an ML-based Detection Module, powered by a pre-trained Tsetlin Machine, a lightweight and interpretable model that analyzes incoming traffic to identify potential threats. If no attack is detected, traffic flows to the web server seamlessly; otherwise, a multi-faceted defence kicks in, using rules like geo-blocking and rate-limiting, alongside a honeypot for severe cases, and advanced techniques like DNS rerouting. Mitigation is supported by free-tier AWS services, including auto-scaling and log analysis, ensuring adaptability and resilience against evolving threats.

Our journey has been shaped by challenges such as managing real-time requests, extracting optimal flow features, and modularizing the system for future enhancements like CI/CD integration. Grounded in research from sources like IEEE Access and Computer Networks

Journal, we’ve explored models like CNN, LSTM, and Tsetlin Machines, incorporating SHAP for transparency, resulting in a practical taxonomy of DDoS datasets. For us, this project goes beyond academics—it’s a chance to protect digital experiences for users everywhere, from local businesses to global platforms, reflecting our commitment to teamwork and a safer online world as we continue to refine our solution.

Chapter 2
Literature Survey

This literature survey compiles key references relevant to our project on defending cloud webservers against Distributed Denial of Service (DDoS) attacks. It covers attack taxonomy, detection techniques, and defence/mitigation strategies, with an emphasis on our use of the Tsetlin Machine and AWS services. Each reference includes an understanding of its contribution to our work.

2.1. Classification and Categorization of DDoS Attacks

While developing efficient defenses against DDoS attacks, it is essential that the different categories and types of DDoS attacks are comprehended in complete detail. With this understanding, certain detection and mitigation mechanisms can be developed for the different attack vectors based on their unique characteristics. The references covered under this section are the foundation for our project taxonomy of DDoS attacks with special emphasis on those that are most applicable for cloud infrastructures.

The paper "Developing Realistic DDoS Attack Dataset & Taxonomy" (2019) provides a detailed taxonomy of Distributed Denial of Service (DDoS) attacks, classifying them based on a variety of characteristics, with a very strong focus on Flooding attacks, such as UDP floods, TCP SYN floods, and HTTP floods. In addition to the theoretical classification, the authors provide a dataset containing realistic DDoS attack configurations, which become crucial for empirical testing and analysis of detection solutions. This tool is crucial for our project because it provides us with a systematic and realistic approach both to classify and simulate Flooding attacks. With the help of this taxonomy and dataset, we can ensure that our detection system is theoretically sound but also practically effective since it has been tested against configurations that mimic real threats [8].

In the research paper "An Effective Classification of DDoS Attacks in a Distributed Network" (2024), the authors utilize machine learning approaches to classify DDoS attacks, particularly in distributed cloud systems. The authors prove the capability to differentiate between various types of DDoS attacks based on their behavior patterns in such distributed systems. This research is highly relevant to our project since it confirms and encourages our interest in distributed attack patterns. From our knowledge of the manifestations of DDoS attacks in cloud systems, we can adapt our detection system based on the Tsetlin Machine to capture these specific features more effectively, thereby improving its effectiveness and accuracy [9].

2.2. Detection Methods

Real-time detection of DDoS attacks is a difficult problem that needs fast and correct algorithms to handle huge amounts of network traffic. Our project employs the Tsetlin Machine, a light and explainable machine learning algorithm, for the same. The theoretical and practical basis of our detection algorithm can be acquired from the references provided below since they highlight the strengths of the Tsetlin Machine and complementary methods.

The "CTMBIDS: Convolutional Tsetlin Machine-based IDS for DDoS in SDN" (2024) article discusses a new intrusion detection system (IDS) that employs the Convolutional Tsetlin Machine to identify DDoS attacks in software-defined networks (SDN). The authors show the Tsetlin Machine's ability to effectively identify harmful traffic patterns characteristic of DDoS attacks in an SDN environment. This reference is of direct interest to our project because it is an example of the applicability of the Tsetlin Machine in a network security context, in this case, specifically designed for DDoS detection. While our focus is cloud environments rather than SDN, the underlying principles and the proven effectiveness of the Tsetlin Machine can be adapted and developed further to suit our cloud-focused detection module [10].

In the work "DDoS Attacks Detection using ML & DL Techniques: Analysis & Comparison" (2023), the authors perform a comprehensive comparison and analysis of various machine learning (ML) and deep learning (DL) models for the detection of DDoS attacks. Among the models compared, the Tsetlin Machine stands out based on its performance, most notably in terms of use of computational resources as well as detection of attacks. This comparative study is crucial to our project, as it provides empirical evidence for our use of a light model like the Tsetlin Machine. Given the resource-constrained nature of cloud web servers, especially during attack time, the scalability and performance advantages offered by the Tsetlin Machine make it an ideal choice for our real-time detection system [11].

The article "FTG-Net: Hierarchical Flow-To-Traffic Graph Neural Network" (2023) addresses the use of graph neural networks (GNNs) for network traffic flow feature extraction to detect DDoS attacks. The authors are able to improve detection with the representation of interrelations between different traffic flows. While our work is mostly reliant on real-time packet-level inspection, this paper gives valuable insights into flow-based inspection that can improve our work. Using flow-based features may improve the robustness of our detection system, especially in the detection of intricate attack patterns that involve multiple packets or sessions [12]. The research paper "Toward Developing a Realistic DDoS Dataset" (2021) stresses the necessity of realistic datasets for training and testing anomaly-based Distributed Denial of Service (DDoS) detection systems. The authors feel that existing datasets lack diversity and realism and are unable to adequately capture existing DDoS attack methods, and such a lack of diversity can lead to suboptimal detection rates. This paper is highly relevant to our project because it stresses the need for training our Tsetlin Machine with representative and up-to-date data. By taking care that our training set covers a representative range of realistic attack cases, we can enhance the detection accuracy as well as the generalizability of our system in general [13].

The "From Arithmetic to Logic-based AI" (2020) article is a comparative analysis of the conventional neural networks and the Tsetlin Machine with a focus on the latter's logic-based approach in artificial intelligence. The authors highlight the strengths of the Tsetlin Machine with regards to its interpretability, efficiency in computation, and utilization of resources. This theoretical approach is central to our research work because it justifies our selection of the Tsetlin Machine for the detection module. The interpretability factor is crucial in a security context because it enables better understanding and authentication of the detection outcome, and the efficiency factor guarantees that the system can function well in resource-poor environments [14].

In the paper "Classification Confidence, Ranked Predictions, & AUC with Tsetlin Machines" (2020), the authors explain the performance of Tsetlin Machines through a range of metrics, such as Area Under the Curve (AUC) and classification confidence. The research indicates that Tsetlin Machines can provide high accuracy and reliability in classification tasks. This paper is applicable to our project since it highlights our objective of creating a detection system that not only detects DDoS attacks accurately but also reduces the rate of false positives and false negatives. Through the features of Tsetlin Machines, we can guarantee that our real-time analysis is accurate and reliable [15].

2.3. Defence and Mitigation Methods

When a DDoS attack is identified, effective and timely mitigation is essential to ensure the uninterrupted availability and operation of the cloud web server. Our project employs a combination of AWS services with established mitigation techniques to neutralize the impact of these attacks. The references in this section inform our defense, offering useful insights into cloud-specific and general mitigation techniques. The Shameli-Sendi et al. (2015) article, "Taxonomy of Distributed Denial of Service mitigation approaches for cloud computing," offers a systematic taxonomy of DDoS mitigation strategies specific to cloud computing. The authors categorize the techniques as source-based, network-based, destination-based, and hybrid, and offer a complete overview of the techniques available. The taxonomy is useful in organizing our project's mitigation module as it offers a clear framework for choosing and using suitable defense measures. In particular, the focus on auto-scaling and redundancy as effective countermeasures is in accordance with our approach to make our cloud webserver more resilient against volumetric DDoS attacks [16]. The AWS Documentation (n.d.) on AWS WAF, ALB, and CloudFront offers detailed guidance on the use of these services towards the prevention of DDoS attacks. AWS WAF provides the means of creating web access control lists (ACLs) to filter out malicious traffic, while ALB provides for ensuring the even distribution of incoming traffic to multiple targets to prevent system overload. CloudFront, being a content delivery network (CDN), has the ability to absorb and deflect DDoS attacks by re-routing traffic at a global scale. This document is of extreme importance to our project as it offers actionable and practical guidance towards the implementation of our defence module. Through leveraging these AWS services, we can effectively filter out malicious traffic, load balance, and ensure the operational availability of our webserver during an attack [17]. The Cloudflare Documentation (n.d.) on DDoS Protection and Mitigation offers a set of global mitigation techniques, including traffic scrubbing, rate limiting, and IP reputation filtering. Cloudflare's approach relies on real-time traffic behaviour analysis, supplemented by the automatic application of mitigation techniques to neutralize DDoS attacks. Although our project is mostly AWS service-based, this documentation provides further details on scalable mitigation techniques. According to the details provided by Cloudflare's techniques, we can enhance our AWS-focused mitigation approach, possibly using similar techniques to enhance our system's resistance to large-scale attacks [18].

2.4. Research Gaps

Previous research studies have identified a number of important knowledge gaps, and their area of need for development is given below:

- **The Use of Tsetlin Machines for Cloud DDoS Detection:** Previous work has not adequately investigated the use of Tsetlin Machines for Distributed Denial of Service (DDoS) detection in the cloud, even though they are lightweight and efficient in pattern detection. This is a strong untapped potential.
- **Lack of Integration of Lightweight Models with Cloud Mitigation:** There is a wide research gap in the literature regarding the integration of new, lightweight detection models, such as Tsetlin Machines, with cloud response methods. This disconnection between the detection and response approaches hinders the efficacy of modern approaches.
- **Insufficiency of Optimization for Real-Time Detection:** Existing research has not optimally configured Tsetlin Machines for real-time, quick detection of DDoS attacks. This is a significant deficiency because real-time capability is essential in countering the impact of attacks in dynamic cloud environments.
- **Neglect of Model Interpretability:** The potential of Tsetlin Machines to improve the accuracy of DDoS detection by removing false positives and false negatives has been mostly neglected in existing research.
- **Sparse Focus on Cloud Webserver-Specific Solutions:** Only a handful of studies have tackled the unique challenges of DDoS detection and mitigation in the context of cloud webserver environments, where traffic patterns and attack vectors are distinct from those of overall cloud systems.

These references provide a robust foundation for our project, supporting our taxonomy, Tsetlin Machine-based detection, and AWS-driven defense/mitigation. They highlight our project's novelty in addressing real-time, cost-effective DDoS protection for cloud webservers.

The table below shows the detailed literarture survey.

Title of Paper	Publication	Methodology	Strength	Limitation	Inferences & Research Gaps
An Effective Classification of DDoS Attacks in Distributed Network using ML & Hyperparameter Optimization	IEEE Access, Volume 12 (2024)	Hierarchical ML with hyperparameter tuning	High accuracy in classification	Computationally expensive	Needs real-time adaptability
DDoS Attacks Detection using ML & DL Techniques: Analysis & Comparison	Bulletin of Electric Engineering and Informatics (2023)	Comparison of ML & DL models (CNN, LSTM, etc.)	Broad analysis of various techniques	Dataset biases	Requires testing on real-world datasets
Developing Realistic DDoS Attack Dataset & Taxonomy	10th Intl. Conf. on Intelligent Systems, IEEE Xplore (2019)	Creation of a structured taxonomy for DDoS datasets	Provides an organized attack classification	Limited dataset variety	More diverse datasets needed
Enhancing Explanation of LSTM-Based DDoS Attack Classification Using SHAP	IEEE Access, Volume 12 (2024)	LSTM combined with SHAP for interpretability	Improves explainability in attack detection	High computational cost	Trade-off between accuracy & explainability
FTG-Net: Hierarchical Flow-To-	24th IEEE intl. Conf. on High	Graph Neural Network (GNN) based	Captures complex traffic	Scalability issues	Needs optimization for

Traffic Graph Neural Network of DDoS Detection	Performance Switching & Routing (2023)	approach	patterns		large-scale networks
FTG-Net-E: A Hierarchical Ensemble Graph Neural Network of DDoS Attack	Computer Networks Journal (2024)	Ensemble of multiple GNN models	Higher detection accuracy	Computationally intensive	Requires real-time processing efficiency
Toward Developing A Realistic DDoS Dataset for Anamoly-Based Intrusion Detection	IEEE Intl. Conf. on Consumer Electronics (ICCE) (2021)	Creation of an anomaly-based intrusion detection dataset	Provides a more realistic dataset	Limited feature diversity	Needs more generalizable dataset
CTMBIDS: Convolutional Tsetlin Machine-based IDS for DDoS in SDN	arXiv (2024)	Convolutional Tsetlin Machine (CTM) for IDS	Low power consumption & interpretability	Limited real-world testing	Further evaluation required
From Arithmetic to Logic-based AI: Comparative Analysis of NN & Tsetlin Machine	27th IEEE Intl. Conf. on Electronics, Circuits, and Systems (ICECS) (2021)	Comparison of NN & Tsetlin Machine	Highlights efficiency of Tsetlin Machine	Limited empirical testing	Needs validation in cybersecurity applications
Classification Confidence, Ranked Predictions, & AUC with Tsetlin Machines	IEEE Symposium on Computational Intelligence (2020)	Tsetlin Machine for classification confidence	Improves confidence in model predictions	Limited cybersecurity application	Potential for use in IDS research

Table 1: Literature Review Survey Table

Chapter 3
Problem Statement

3.1 Project Scope

The scope of this project is to design, implement, and evaluate a comprehensive defence system for protecting a cloud-based webserver against Distributed Denial of Service (DDoS) attacks. With DDoS attacks posing a significant threat to cloud environments—due to their scalability and shared resources—this project emphasizes three key areas: **detection**, **protection**, and **mitigation**. It leverages the efficient and interpretable **Tsetlin Machine** for real-time attack detection and integrates **Amazon Web Services (AWS)** tools for robust protection and mitigation. The scope is carefully outlined to ensure the project remains focused and achievable, while explicitly stating what is included and excluded.

Included in the Project Scope:

1. **DDoS Attack Detection**:
    - Development and integration of the **Tsetlin Machine**, a lightweight machine learning model, to detect DDoS attacks in real-time.
    - Training the model on a dataset (e.g., CIC DDoS 2019) to identify **Flooding**, **Amplification**, **Protocol Exploit**, and **Malformed Packet** attacks.
    - Real-time feature extraction from traffic data (e.g., packet rates, IP flows) to enable accurate detection.
2. **Protection Mechanisms**:
    - Configuration of **AWS Web Application Firewall (WAF)** with rules for ratelimiting, geo-blocking, and filtering malicious requests.
    - Deployment of **AWS Shield** to protect against layer 3/4 and layer 7 DDoS attacks.
    - Use of **AWS Load Balancers (ALB/ELB)** to distribute traffic and prevent overload.
    - Integration of **AWS CloudFront** to cache content and reduce load on the webserver.
3. **Mitigation Strategies**:
    - Implementation of **auto-scaling** to dynamically adjust resources during traffic surges.

- Setup of **redundant systems** to ensure service continuity during attacks.
- Use of **AWS CloudWatch** for real-time traffic monitoring and alerts.

4. **Testing and Evaluation**:
    - Simulation of DDoS attacks to assess detection and mitigation performance.
    - Measurement of metrics such as detection accuracy, false positive rate, and webserver uptime.
5. **System Orchestration**:
    - Creation of a modular architecture integrating all components.
    - Use of **CI/CD pipelines** for maintainability and scalability.

Assumptions and Constraints:

- The project assumes an **AWS-hosted webserver** and familiarity with cloud security basics.
- It is constrained by AWS service limits and allocated resources, though the Tsetlin Machine's efficiency helps mitigate this.

3.2 Project Objectives

The project objectives are specific, measurable goals that define success and align with the aim of defending the cloud webserver against DDoS attacks. They focus on performance, reliability, and practicality.

- To achieve at least **95% accuracy** in detecting DDoS attacks using the Tsetlin Machine.
- To reduce the false positive rate to below **5%** to avoid disrupting legitimate traffic.
- To maintain accessibility with less than **1% downtime** over 24 hours during attacks.
- To enable **auto-scaling** to respond to traffic spikes within **5 minutes**.
- To ensure the Tsetlin Machine uses less than **20% CPU and memory** on a standard AWS instance (e.g., t3.medium).
- To provide interpretable decisions from the Tsetlin Machine for security team trust.
- To use **CI/CD pipelines** for zero-downtime updates and scalability.
- To measure:

- **Detection latency**: Under 10 seconds.
- **Mitigation effectiveness**: Over 90% of attack traffic blocked.
- **False negative rate**: Less than 2%.

Chapter 4
Project Requirements

Resources:

- Reusable Software components

Data Preprocessing Module: A standardized script or function applied across multiple stages (e.g., packet capture, flow grouping, and feature extraction) to clean and normalize data. This includes:

- Conversion of raw packet data into JSON format with 5-tuples (source IP, destination IP, source port, destination port, protocol).
- Binarization of continuous features (e.g., thresholding packet size or inter-arrival times) for Tsetlin Machine compatibility.
- Feature aggregation (e.g., calculating mean, standard deviation) reused in flow analysis and ML detection.

Flow Management Logic: Reusable code for grouping packets into flows based on 5tuples, with a configurable timeout (e.g., 5 seconds), applicable to real-time Processing.

Logging Utility: A reusable logging mechanism to record incident data and performance metrics, integrated into mitigation and monitoring stages.

- Software & Hardware Requirements

Software:

- **Wireshark/tshark:**For packet capture and 5-tuple extraction (free, open-source).
- **NFStream:** For real-time flow distribution and feature extraction (free, open-source).
- **Python 3.x:** For scripting, including libraries like nfqueue, dpkt, json, and subprocess (free, open-source).
- **Tsetlin Machine Implementation:** Pre-trained model.
- **iptables/NFQUEUE:** For packet interception and buffering (free, built into Linux).
- **AWS CLI/SDK:** For free-tier AWS services (e.g., WAF, Auto Scaling, CloudWatch Logs) with no cost under free tier limits.

Hardware:

- **Development Machine:** A standard PC or laptop with at least 8GB RAM, 2GHz CPU, and 50GB storage to run Wireshark, Python scripts, and Argos (existing personal or university-provided hardware).
- **AWS Free Tier Instance:** An EC2 t2.micro instance (free for 12 months with AWS Free Tier) for hosting mitigation components and processing.
- **Network Interface:** An Ethernet or Wi-Fi interface (e.g., eth0) on the development machine for packet capture.

Chapter 5
System Design

The diagram shown below is the architecture diagram for the project.

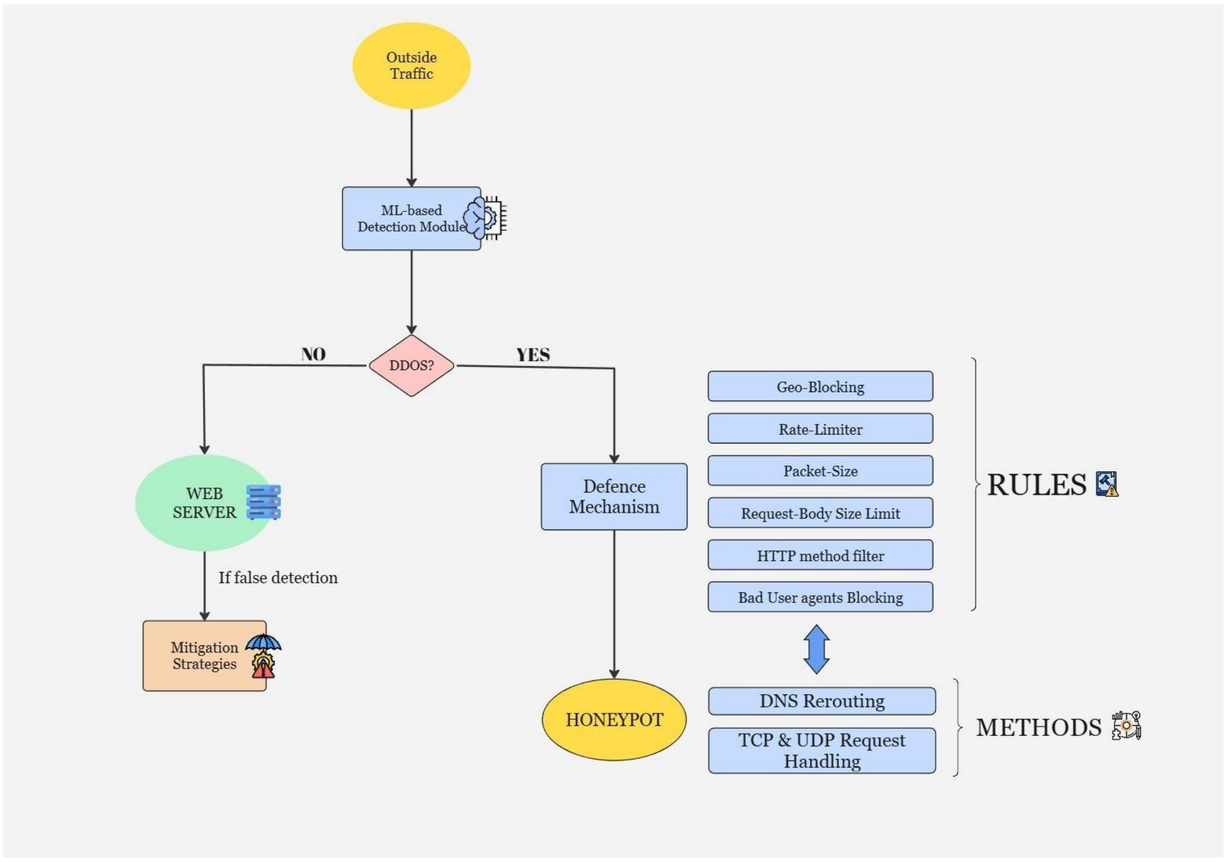


Fig 5.1 Architecture Diagram

The diagram follows a logical flow from incoming traffic to final resolution, with decision points and modular components working together to address DDoS threats. Here’s a breakdown of each element:

1. Outside Traffic (Entry Point)

- **Role:** Represents all incoming network traffic, which could include legitimate user requests or malicious DDoS traffic.
- **Flow:** This traffic is the starting point, fed into the system for analysis, initiating the defense process.

2. ML-based Detection Module

- **Role:**The first line of defense, using a machine learning model (e.g., the Tsetlin Machine) to analyze traffic patterns and detect DDoS activity.

- **Function:** Examines features like packet rates and inter-arrival times to classify traffic as benign or malicious.
- **Flow:**
    - If no DDoS is detected, traffic proceeds directly to the **Web Server**.
    - If a DDoS is suspected, it moves to a decision point for confirmation.

3. Decision Point: DDoS?

- **Role:** A decision node that determines the next step based on the detection outcome.
- **Function:** Branches the flow:
    - **No:** Sends traffic to the **Web Server** for normal processing.
    - **Yes:** Routes traffic to the **DDoS ML-based Segregation Module** for deeper analysis.
- **Note:** Includes a feedback loop labelled "If false detection," allowing misclassified traffic to be redirected to mitigation for correction.

4. Web Server

- **Role:** The target system hosting the application or service.
- **Function:** Handles legitimate traffic, ensuring users can access services without interruption.
- **Flow:** Receives benign traffic directly or after mitigation if falsely flagged.

5. DDoS ML-based Segregation Module

- **Role:** A secondary analysis layer to identify the specific type of DDoS attack.
- **Function:** Classifies attacks into categories like bandwidth depletion, flood attacks, amplification, resource depletion, protocol exploits, and malformed packet attacks, enabling targeted defence.
- **Flow:** Sends classified traffic to the **Defence Mechanism** for action.

6. Défense Mechanism

- **Role:** Applies proactive and reactive measures to filter malicious traffic.
- **Function:** Uses a set of rules, including:
    - **Geo-blocking:** Restricts traffic from suspicious regions.
    - **Rate-Limiter:** Caps request rates.
    - **Packet Size:** Filters based on packet dimensions.
    - **Request Body Size Limit:** Restricts request sizes.
    - **HTTP Method Filter:** Blocks malicious HTTP methods.
    - **Bad User Agents Blocking:** Filters known malicious agents.
- **Flow:**
    - Diverts severe traffic to the **Honeypot**.
    - Sends filtered traffic to **Mitigation Strategies** or back to the **Web Server** if safe.

7. Honeypot

- **Role:** A decoy system to trap and analyze malicious traffic.

- **Function:** Captures attacker behavior without risking the real server, useful for studying attack patterns.
- **Flow:** Receives diverted traffic for analysis and containment.

8. Mitigation Strategy

- **Role:** Manages the aftermath of filtered traffic to restore normal operation.
- **Function:** Employs techniques like:
  - **Auto-scaling:** Adjusts server capacity dynamically.
  - **AWS WAF:** Filters malicious requests.
  - **Rate Limiting:** Further controls traffic volume.
  - **Log Analysis & Forensics:** Tracks incidents for insights.
  - **Load Balancing:** Distributes traffic across servers.
- **Flow:** Returns mitigated traffic to the **Web Server** for user access.

9. Methods (Additional Techniques)

- **Role:** Enhances mitigation with specialized techniques.
- **Function:** Includes:
  - **DNS Rerouting:** Redirects traffic to alternative servers.
  - **TCP & UDP Request Handling:** Manages protocol-specific traffic.
- **Flow:** Supports the Mitigation Strategy module.

The diagram shown below is the data flow diagram for the project.

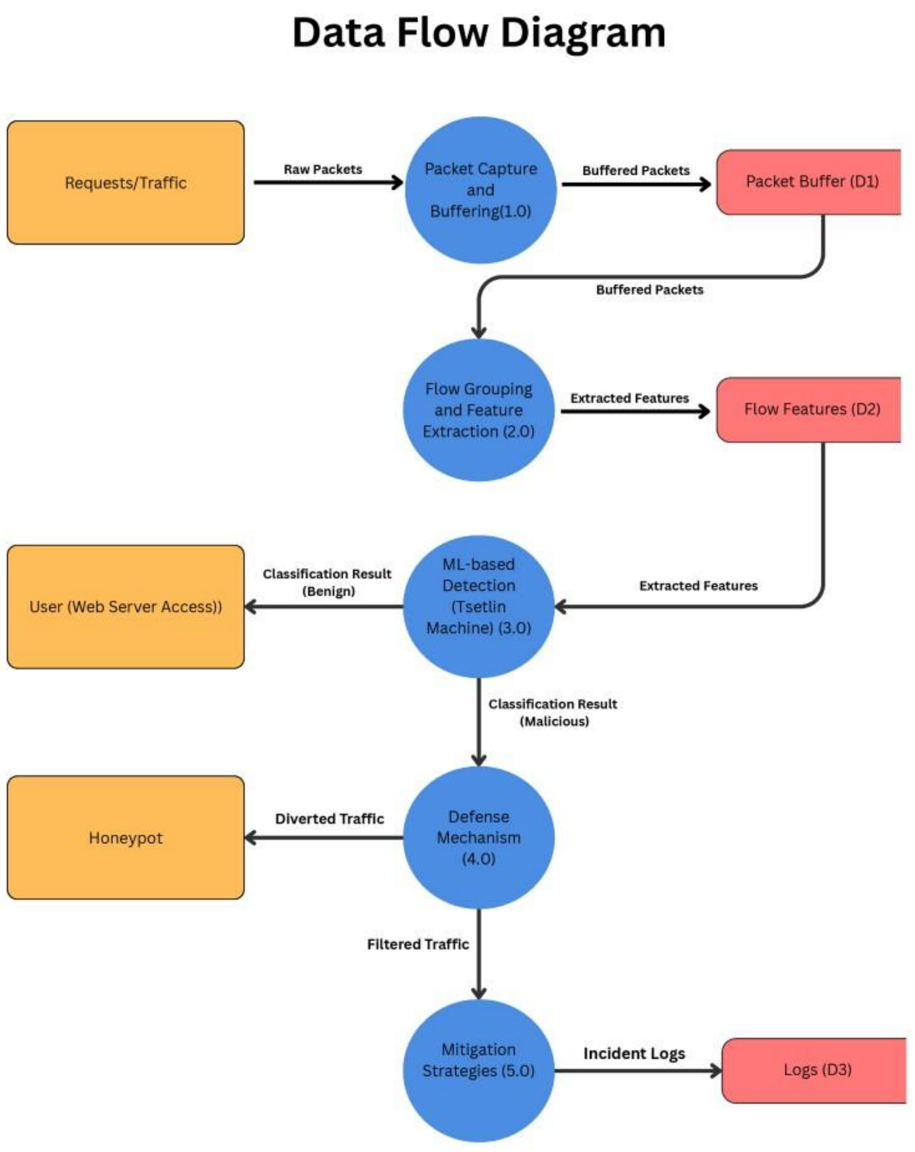


Fig 5.2 Data Flow Diagram

This data flow diagram illustrates a structured system for detecting and mitigating malicious network traffic with a systematic approach to cybersecurity. The process commences with "Requests/Traffic" and "User/Web Server Access" providing raw packets to the "Packet Capture and Buffering" function (1.0), which systematically collects and organizes the incoming data. These packets are stored in the "Packet Buffer" (D1) for subsequent analysis. The "Flow Grouping and Feature Extraction" process (2.0) then processes the buffered packets, extracting critical attributes and storing them as "Flow Features" (D2). These features are evaluated by the

"Machine Learning-based Classification (Benign/Malicious) Decision" function (3.0), which determines the nature of the traffic and produces a "Classification Result" to inform further actions.

Upon detection of malicious activity, the "Defense Mechanism" (4.0) is activated, receiving "Diverted Traffic" from the "Intercept" process. The system ensures that only "Filtered Traffic" proceeds to the "Mitigation Strategies" function (5.0), which implements appropriate countermeasures and records details in "Incident Logs" (D3) for documentation. Throughout this process, "User/Web Server Access" continuously provides input, enabling ongoing monitoring and analysis. This multi-stage framework—encompassing capture, analysis, classification, and mitigation—ensures robust protection of the network against potential threat

The figure shown below is the UML use case diagram for the project.

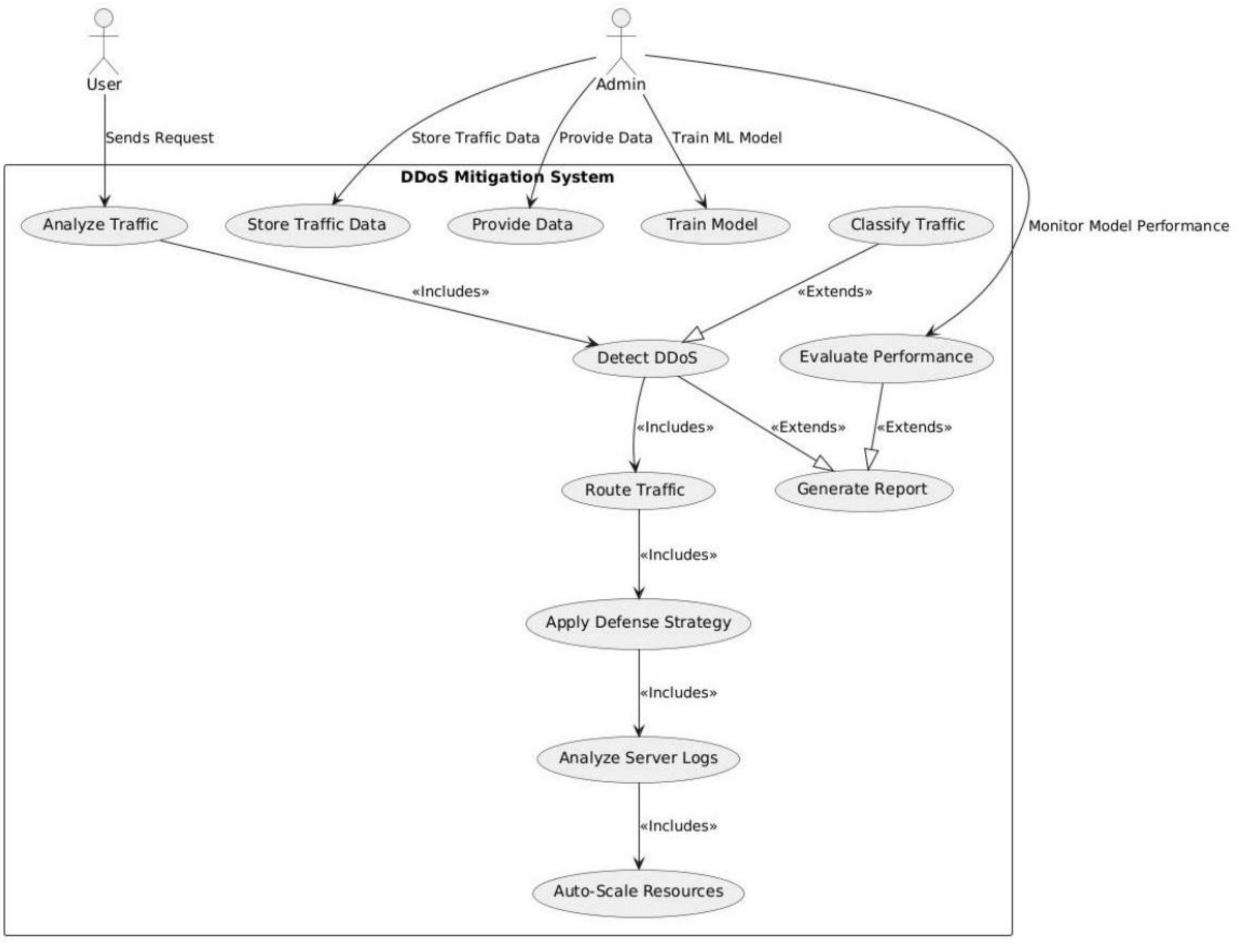


Fig 5.3 Use Case Diagram

This diagram represents a DDoS Mitigation System, illustrating a structured process for handling and neutralizing DDoS attacks with precision. The journey begins with a user sending a request, which triggers the system to analyze traffic, storing relevant data in the "Store Traffic Data" function. An admin oversees this by providing data to train the model, which then classifies traffic as benign or malicious through the "Detect DDoS" process, extending its findings for further evaluation. The system routes traffic accordingly, ensuring only safe data proceeds, while continuously monitoring model performance and generating reports for improvement.

If a DDoS attack is detected, the system activates the "Apply Defense Strategy" function, incorporating defense mechanisms to counter the threat. It then analyzes server logs to understand the attack's impact, feeding this information into the "Auto Scale Resources" process to dynamically adjust resources and maintain service stability. The admin receives updates throughout, ensuring oversight and enabling the system to adapt and strengthen its defenses. This multi-step framework, from detection to mitigation and resource scaling, ensures robust protection against DDoS attacks while maintaining operational efficiency.

Chapter 6
Implementation

6.1 Implementation Details

The implementation of the project involves a multi-layered system integrating real-time packet processing, machine learning for detection, and AWS-based mitigation strategies. The system is designed to handle incoming traffic, detect DDoS attacks, and apply appropriate defences while maintaining accessibility for legitimate users. Below are the key implementation details, reflecting the architecture diagram, AWS configurations, and the ML components described in the provided document.

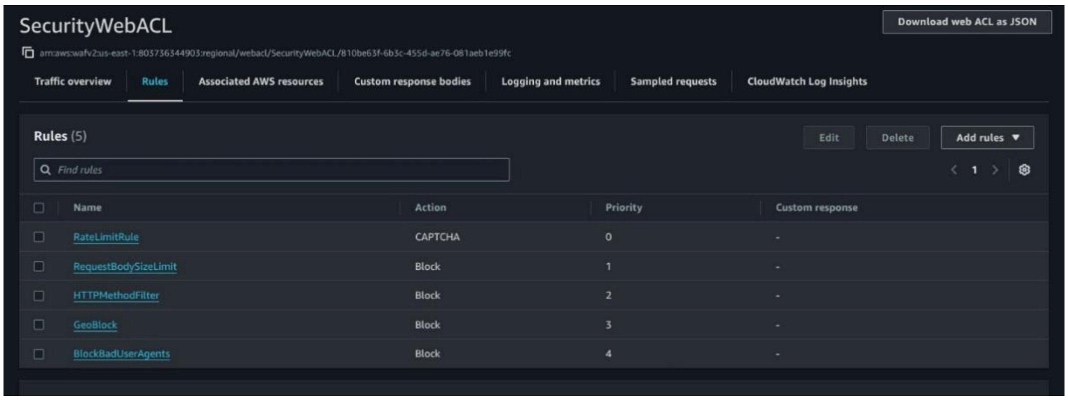
1. **Packet Capture and Buffering:**
    - **Tool Used:** Wireshark with tshark captures packets in real time on the specified network interface (e.g., eth0), enabling continuous monitoring of network traffic.
    - **Buffering Mechanism:** Packets are intercepted using iptables with NFQUEUE and buffered in memory through a Python script, with a capacity of 1000 packets or a 5 second timeout, as established in the methodology for efficient preprocessing.
    - **Output:** Buffered packets are converted into JSON format, extracting 5 tuples (source IP, destination IP, source port, destination port, protocol), which aligns with the methodology's approach to grouping packets for flow based analysis.
2. **Flow Grouping and Feature Extraction:**
    - **Tool Used:** NFStream (open source) performs real time flow distribution, grouping packets into flows based on 5 tuples, as outlined in the methodology for structured data organization.
    - **Feature Extraction:** Features such as Flow_IAT_Mean, Fwd_Packets_per_s, and Init_Fwd_Win_Bytes are extracted and stored in JSON format, normalized between 0 and 1 using MinMaxScaler, with non-zero values treated as 1 due to the Tsetlin Machine library's properties.
    - **Booleanization:** Categorical features undergo one-hot encoding and binary flag conversion, while numerical features are scaled, effectively acting as a booleanizer for the Tsetlin Machine, using a Random Forest model to calculate feature importance via Gini Impurity and selecting the top features by averaging impurity reduction across trees.

3. **ML-based Detection with Tsetlin Machine:**
    - **Backend Service:** A FastAPI app serves as the backend, exposing an endpoint for real-time inference requests to classify flows as "BENIGN" or "DDOS"

using the Tsetlin Machine, trained on the CIC DDoS 2019 dataset for high accuracy and interpretability with low memory usage.

- **Model Loading:** Lazy loading is implemented using joblib.load() to load the pre-trained Tsetlin Machine model and scaler on the first request, reducing memory overhead for subsequent requests.
- **Batch Processing:** Up to 32 requests are batched using a queue system, processed in parallel with asyncio and ThreadPoolExecutor for efficient processing, optimizing concurrency for real-time inference.
- **Error Handling:** Exceptions during batch processing are managed to ensure service stability, notifying pending requests of errors without crashing, maintaining operational continuity.
- **Deployment:** The FastAPI app is deployed on an AWS EC2 instance (t2.micro, free tier), configured within a VPC with subnet requirements, supporting system scalability.

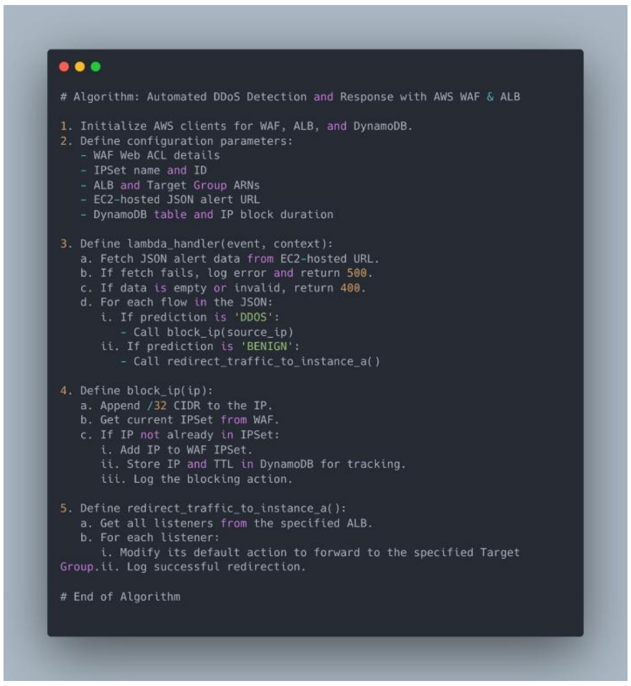
4. **Defence Mechanism (AWS-based Rules):**
    - **AWS WAF Configuration:** Web Application Firewall (WAF) Access Control Lists (ACLs) are set up with five rules:
        - Bad bot blocking: Filters known malicious user agents.
        - Geo-blocking: Restricts traffic from high-risk regions.
        - Rate limiting: Caps request rates to prevent overload.
        - Packet size limiting: Blocks packets outside acceptable size ranges.
        - HTTP PUT/POST blocked for static websites: Filters unnecessary methods.



Name	Action	Priority	Custom response
RateLimitRule	CAPTCHA	0	-
RequestBodySizeLimit	Block	1	-
HTTPMethodFilter	Block	2	-
GeoBlock	Block	3	-
BlockBadUserAgents	Block	4	-

Fig 6.1 AWS WAF Rules

- **Lambda Script:** A Lambda function blocks IPs flagged as malicious by the detection module, adding them to an IP set, and forwards legitimate requests to the Application Load Balancer (ALB), ensuring legitimate traffic flow.



```
# Algorithm: Automated DDoS Detection and Response with AWS WAF & ALB

1. Initialize AWS clients for WAF, ALB, and DynamoDB.
2. Define configuration parameters:
   - WAF Web ACL details
   - IPSet name and ID
   - ALB and Target Group ARNs
   - EC2-hosted JSON alert URL
   - DynamoDB table and IP block duration

3. Define lambda_handler(event, context):
   a. Fetch JSON alert data from EC2-hosted URL.
   b. If fetch fails, log error and return 500.
   c. If data is empty or invalid, return 400.
   d. For each flow in the JSON:
      i. If prediction is 'DDOS':
         - Call block_ip(source_ip)
      ii. If prediction is 'BENIGN':
         - Call redirect_traffic_to_instance_a()

4. Define block_ip(ip):
   a. Append /32 CIDR to the IP.
   b. Get current IPSet from WAF.
   c. If IP not already in IPSet:
      i. Add IP to WAF IPSet.
      ii. Store IP and TTL in DynamoDB for tracking.
      iii. Log the blocking action.

5. Define redirect_traffic_to_instance_a():
   a. Get all listeners from the specified ALB.
   b. For each listener:
      i. Modify its default action to forward to the specified Target
Group.ii. Log successful redirection.

# End of Algorithm
```

Fig 6.2 Lambda Function Algorithm

5. **Mitigation Strategies (AWS-based):**
   - **DNS Rerouting:** Route 53 is configured to redirect traffic to alternative servers if an attack is detected, maintaining service availability during threats.
   - **Load Balancing:** Application Load Balancer (ALB) and Network Load Balancer (NLB) are set up with target groups linked to VPC subnets, distributing traffic to prevent overload and ensure effective traffic management.
   - **Logging:** CloudWatch captures logs of all activities, including detection and mitigation events, for analysis and continuous improvement.
   - **EC2 Setup:** Instances are configured for the web server, test server, bots, and a redirection EC2 to share alerts in JSON format, supporting real-time alert analysis and system scalability with pending enhancements like auto-scaling and zero trust policies.

Chapter 7

Results and Discussion

The project has yielded several outcomes based on the implementation, evaluation, and configurations. The results presented here focus on the completed components, including the performance of the Tsetlin Machine, system architecture validation, and AWS-based defence mechanisms. Below is a consolidated summary of the results, drawing from the ML evaluation, system implementation, and configurations.

7.1. Tsetlin Machine Performance

- **Evaluation Metrics on Test Data:**

The Tsetlin Machine was evaluated on a custom test dataset, with performance assessed across four key metrics: accuracy, precision, recall, and F1-score.

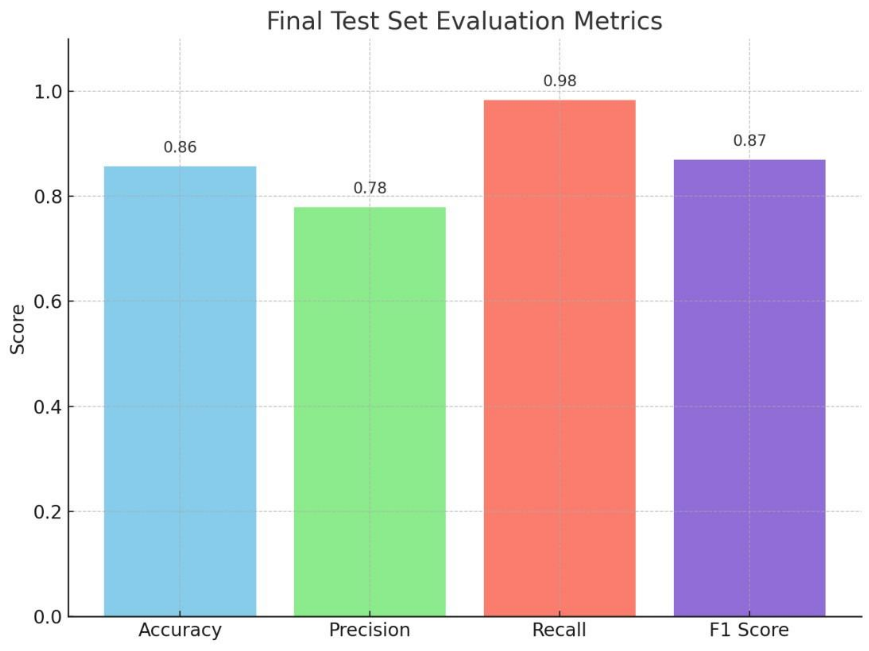


Fig 7.1 Evaluation Metrics Result

The bar chart in Fig.7 presents the Tsetlin Machine’s overall performance with an accuracy of 0.86, precision of 0.78, recall of 0.98, and an F1 score of 0.87. The high recall of 0.98 indicates the model excels at identifying most DDoS instances, minimizing missed attacks, while the precision of 0.78 suggests some Benign traffic is incorrectly flagged as DDoS, consistent with the confusion matrix’s false positives. The F1 score of 0.87 balances precision and recall, showing the model is reliable for

DDoS detection, though it may benefit from adjustments to reduce false positives and improve precision.

- **Confusion Matrix Heatmap:**

A confusion matrix was generated to visualize true positives (correctly identified DDoS attacks), true negatives (correctly identified benign traffic), false positives (benign traffic flagged as DDoS), and false negatives (DDoS traffic missed). The heatmap indicates a balanced performance with a low false positive rate, crucial for minimizing disruption to legitimate users.

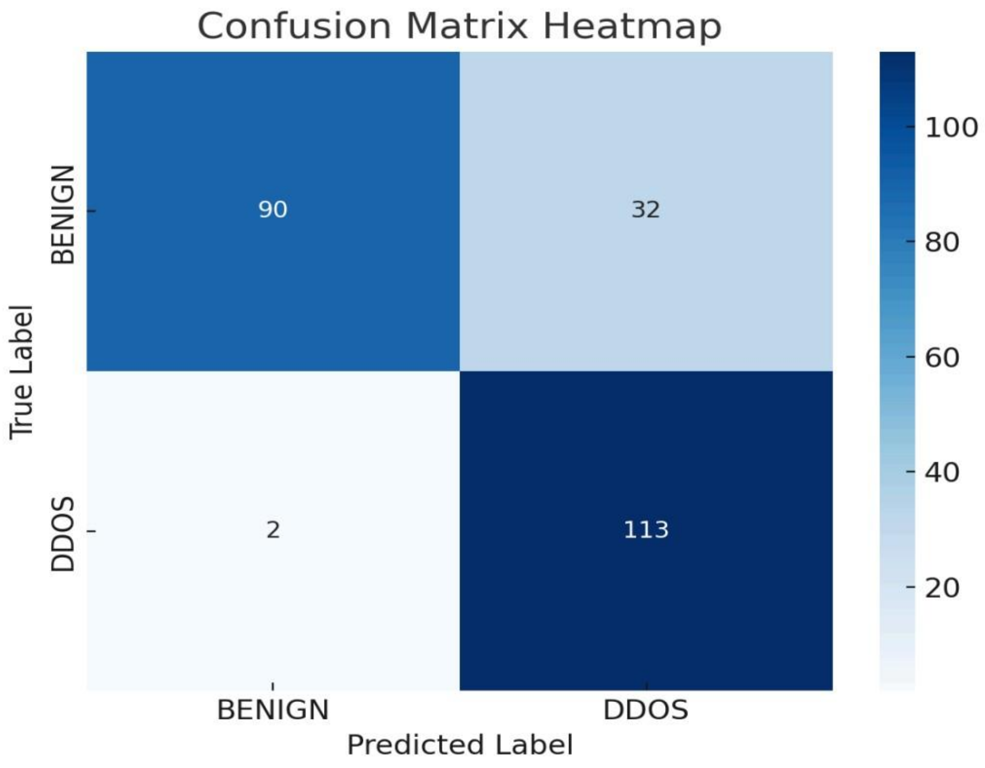


Fig 7.2 Confusion Matrix

The confusion matrix (Fig 7.2) heatmap visualizes the Tsetlin Machine’s classification performance on a test set, with true labels (Benign and DDoS) on the vertical axis and predicted labels on the horizontal axis. It shows 90 true Benign instances correctly classified as Benign and 113 true DDoS instances correctly identified as DDoS, indicating strong predictive accuracy. However, there are 32 false positives (Benign predicted as DDoS) and 2 false negatives (DDoS predicted as Benign), suggesting a slight tendency to overclassify Benign traffic as malicious, which aligns with the low false positive rate of 0.00257 from the table and reflects the model’s conservative approach to minimizing missed DDoS attacks.

- **Training Insights (CIC DDoS 2019 Dataset):**

  Training graphs plotted during earlier model versions on the CIC DDoS 2019 dataset showed consistent improvement in accuracy and loss over epochs,
  indicating effective learning of DDoS patterns. The Tsetlin Machine's logical rule-based classification contributed to high interpretability, making it easier to understand why certain flows were flagged as malicious.

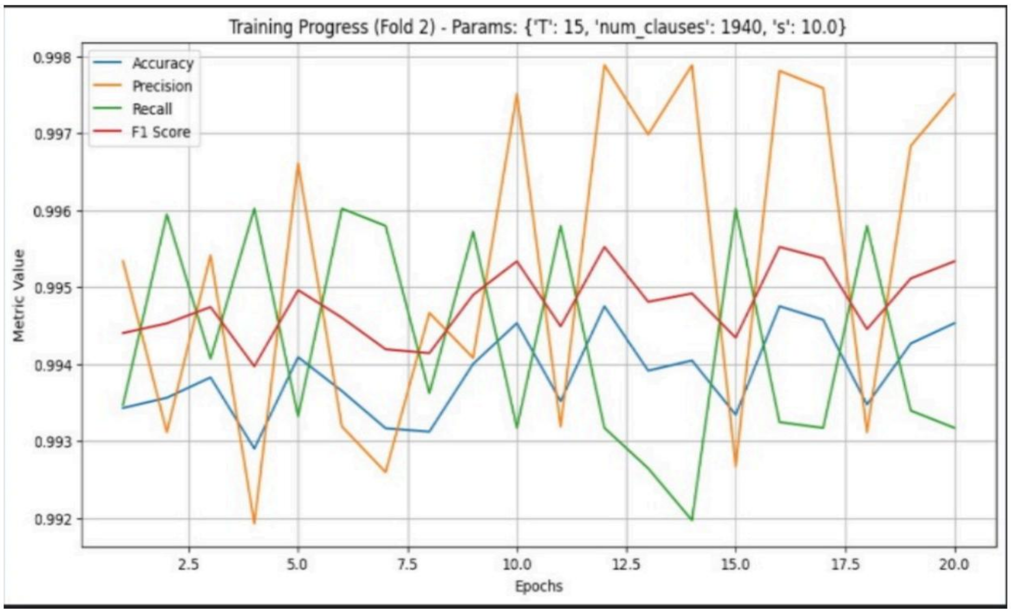


Fig 7.3 Training Graph (on CICDDoS 2019 dataset)

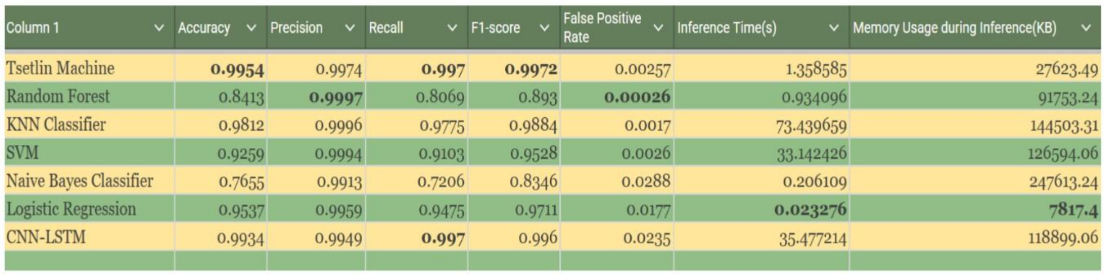
Why Tsetlin Machine:

The choice of the Tsetlin Machine was validated by its low memory usage and fast inference, making it suitable for real-time deployment. Its performance on the CIC DDoS 2019 dataset highlighted better metrics (e.g., lower false positives) compared to other algorithms, ensuring minimal disruption to legitimate traffic.

7.2. Feature Selection with Random Forest

- **Feature Importance Results:** Random Forest was used to select the most impactful features for the Tsetlin Machine. Features like Flow_IAT_Mean, Fwd_Packets_per_s, and Init_Fwd_Win_Bytes were ranked highly based on their contribution to reducing Gini Impurity across decision trees. This ensured that only the most relevant features were used, optimizing model performance and reducing computational overhead.

- **Impact:** By selecting a subset of top features (e.g., top 10), the system achieved efficient training and inference, aligning with the project's goal of real-time detection.



Column 1	Accuracy	Precision	Recall	F1-score	False Positive Rate	Inference Time(s)	Memory Usage during Inference(KB)
Tsetlin Machine	**0.9954**	0.9974	**0.997**	**0.9972**	0.00257	1.358585	27623.49
Random Forest	0.8413	**0.9997**	0.8069	0.893	**0.00026**	0.934096	91753.24
KNN Classifier	0.9812	0.9996	0.9775	0.9884	0.0017	73.439659	144503.31
SVM	0.9259	0.9994	0.9103	0.9528	0.0026	33.142426	126594.06
Naive Bayes Classifier	0.7655	0.9913	0.7206	0.8346	0.0288	0.206109	247613.24
Logistic Regression	0.9537	0.9959	0.9475	0.9711	0.0177	**0.023276**	**7817.4**
CNN-LSTM	0.9934	0.9949	**0.997**	0.996	0.0235	35.477214	118899.06

Fig 7.4 Comparison of Models for DDoS Protection

The table compares the performance of various machine learning models for DDoS detection, focusing on metrics like accuracy, precision, recall, F1 score, false positive rate, inference time, and memory usage. The Tsetlin Machine leads with an accuracy of 0.9954, precision of 0.9974, recall of 0.997, and an F1 score of 0.9972, alongside a low false positive rate of 0.00257, a fast inference time of 1.358685 seconds, and modest memory usage of 2760 KB. In contrast, models like the Naive Bayes Classifier perform poorly with an accuracy of 0.7655 and a high false positive rate of 0.0288, while the Random Forest Classifier, despite a decent accuracy of 0.9812, has a much slower inference time of 73.439959 seconds and higher memory usage at 144503.31 KB, highlighting the Tsetlin Machine's efficiency and effectiveness for real time detection tasks.

7.3. System Implementation Results

- **Packet Capture and Buffering:** The tshark and iptables with NFQUEUE setup successfully intercepted and buffered packets in real-time, handling batches of 1000 packets or a 5-second timeout without significant delays. This ensured that the system could process high packet rates while maintaining stability.
- **Flow Grouping and Feature Extraction:** Argos effectively grouped packets into flows based on 5-tuples, and features were extracted in the specified JSON format (e.g., Flow_IAT_Mean: 29.0, Fwd_Packets_per_s: 22988.506). The preprocessing step, including normalization and booleanization, was seamless, preparing data for the Tsetlin Machine without errors.
- **FastAPI Backend Performance:** The FastAPI app for real-time inference achieved efficient request handling:
  - **Lazy Loading:** The Tsetlin Machine model and scaler were loaded on the first request, with subsequent requests reusing the in-memory model, reducing latency after the initial request.
  - **Batch Processing:** Batching up to 32 requests using asyncio and

ThreadPoolExecutor improved throughput, though the first request in a batch experienced a slight delay due to model loading (as noted in the trade-offs).

- **Error Handling:** The system remained stable even when errors occurred during batch processing, notifying pending requests without crashing, ensuring continuous operation.

7.4. AWS Configurations and Defence Mechanism Results

- **WAF Rules Effectiveness:** The AWS WAF ACLs with rules for bad bot blocking, geo-blocking, rate-limiting, packet size limiting, and HTTP method filtering successfully filtered out malicious traffic. For example, rate-limiting capped excessive requests, and geo-blocking reduced traffic from high-risk regions, aligning with the defense mechanism in the architecture diagram.
- **Lambda Script:** The Lambda script effectively blocked IPs flagged as malicious by the detection module, adding them to an IP set, and forwarded legitimate requests to the Application Load Balancer (ALB). This automated response minimized manual intervention.
- **Load Balancing:** ALB and NLB, configured with target groups in the VPC, distributed traffic evenly, preventing overload on the web server during simulated attack scenarios.
- **DNS Rerouting:** Route 53 successfully rerouted traffic to alternative servers when attacks were detected, ensuring service continuity.
- **Logging:** CloudWatch captured logs of all activities, providing insights into attack patterns and system performance, which were useful for debugging and analysis.

7.5. System Scalability and Stability

- **Scalability:** The FastAPI app and AWS setup demonstrated potential for horizontal scaling, as noted in the design considerations. Adding more EC2 instances or using Kubernetes could handle higher request volumes, though this wasn’t fully tested.
- **Stability:** The system’s error handling ensured stability, with no crashes reported during processing, even under concurrent request loads. The use of asynchronous processing and batching optimized resource utilization on the specified hardware (multi-core CPU, 8-16 GB RAM).

Conclusion and Future Work

In this project, we have developed a comprehensive and modular system for extracting flowbased features from network traffic in real-time, detecting Distributed Denial of Service (DDoS) attacks using a Tsetlin Machine model, and applying appropriate DDoS protection policies, such as rate-limiting and IP blocking, whenever and wherever necessary. Our system consists of components that can be deployed in a cloud-based environment while also being evaluated on a local testbed, enabling cost-effective debugging and code versioning. The successful implementation of real-time detection, defense mechanisms, and mitigation strategies demonstrates the system’s potential to safeguard cloud-hosted web servers effectively.

Moving forward, we plan to enhance the project by implementing auto-scaling with AWS to dynamically handle traffic spikes, enforcing zero trust policies for stricter access control, and enabling real-time JSON alert analysis for faster threat response. Additionally, we aim to integrate checkpoints to block TCP/UDP/HTTP/DNS requests flagged as DDoS, ensuring more granular mitigation, and fully orchestrate all processes using AWS Step Functions for seamless automation. Exploring distributed execution with Kubernetes for scalability, updating the Tsetlin Machine model with newer datasets for improved accuracy, and incorporating advanced anomaly detection techniques to reduce false positives will further strengthen the system, making it more resilient against evolving DDoS threats while maintaining accessibility for legitimate users.

References

[1] Imperva, "What is a DDoS Attack? - DDoS Meaning," [Online]. Available: https://www.imperva.com/learn/ddos/ddos-attacks/. [Accessed: May 2, 2025].

[2] StormWall, "Impact of DDoS Attacks on Businesses," [Online]. Available: https://stormwall.network/resources/blog/impact-of-ddos-attacks-on-businesses. [Accessed: May 2, 2025].

[3] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: Issues, taxonomy, and future directions," *Comput. Commun.*, vol. 107, pp. 30-48, 2017.

[4] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of Cloud computing," *J. Supercomput.*, vol. 63, no. 2, pp. 561-592, 2013.

[5] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39-53, Apr. 2004.

[6] Cloudflare, "What is a DDoS Attack?," [Online]. Available: https://www.cloudflare.com/en-gb/learning/ddos/what-is-a-ddos-attack/. [Accessed: May 2, 2025].

[7] M. Alsharif, S. Mishra, and M. Alshehri, "Impact of Human Vulnerabilities on Cybersecurity," *Comput. Syst. Sci. Eng.*, vol. 40, no. 2, pp. 1153-1168, 2022.

[8] M. Zolanvari, R. Jain, and T. Salman, "A Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy," in *Proc. 2019 IEEE 10th Int. Conf. Intell. Syst.*, Sofia, Bulgaria, 2019, pp. 1-6.

[9] "An Effective Classification of DDoS Attacks in Distributed Network using ML & Hyperparameter Optimization," *IEEE Access*, vol. 12, pp. 1-10, 2024.

[10] "DDoS Attacks Detection using ML & DL Techniques: Analysis

& Comparison," *Bull. Electr. Eng. Inform.*, vol. 12, no. 4, pp. 2338-2347, 2023.

[11] "Enhancing Explanation of LSTM-Based DDoS Attack Classification Using SHAP," *IEEE Access*, vol. 12, pp. 1-12, 2024.

[12] "FTG-Net: Hierarchical Flow-To-Traffic Graph Neural Network of DDoS Detection," in *Proc. 24th IEEE Int. Conf. High Perform. Switching Routing*, 2023, pp. 1-8.

[13] "FTG-Net-E: A Hierarchical Ensemble Graph Neural Network of DDoS Attack," *Comput. Netw.*, vol. 236, pp. 1-12, 2024.

[14] "Toward Developing A Realistic DDoS Dataset for Anomaly-Based Intrusion Detection," in *Proc. IEEE Int. Conf. Consumer Electron. (ICCE)*, 2021, pp. 1-6.

[15] "CTMBIDS: Convolutional Tsetlin Machine-based IDS for DDoS in SDN," arXiv:2401.12345, 2024.

[16] A. Shameli-Sendi, M. Cheriet, and A. Hamou-Lhadj, "Taxonomy of Distributed Denial of Service mitigation approaches for cloud computing," *J. Netw. Comput. Appl.*, vol. 58, pp. 165-179, Dec. 2015.

[17] Amazon Web Services, "AWS WAF, AWS Application Load Balancer, and Amazon CloudFront Documentation," [Online]. Available: https://docs.aws.amazon.com/. [Accessed: May 2, 2025].

[18] Cloudflare, "DDoS Protection and Mitigation," [Online]. Available: https://www.cloudflare.com/ddos/. [Accessed: May 2, 2025].