Stardom University

Stardom Scientific Journal of Natural and Engineering Sciences

- Stardom Scientific Journal of Natural and Engineering Sciences -
Peer Reviewed Scientific Journal published twice
a year by Stardom University

2nd issue- 4th Volume 2026

ISSN 3756-2980

Editorial Team

Editor-in-Chief

Prof. Sayed Hemeda — Egypt

Editorial Board Manager

Assoc. Prof. Dr. Redwan Mohammed Saad — Yemen

Proofreader

Dr. Basem Al-Faqeer - Jordan

Editorial Board Members

Professor, Chongqing University — China

Professor Antonio Zanutta — Italy

Assoc. Prof. Amin Beiranvand Pour — Malaysia

Prof. Maria Stefanidou — Greece

Dr. Ahmed Mohsen Hassan Metwally — Egypt

Alaa Ibrahem Mohamed Ali — Egypt

Prof. Elsadig Arbab Hagar — Sudan

Asso. Prof Salem Barahma — Yemen

Head of Consultancy Committee

Dr. Taha A. Elwi — Iraq

All literary and artistic property rights are reserved for Stardom Scientific Journal of Natural and Engineering Sciences

Securing Load Balancers Against AI and Protocol Threats

Prepared by
Res: Shadi Radwan Albasla

Supervised by
Dr. Nejood Hashim Al-Walidi

Department of IT (Cybersecurity)

Stardom University

Abstract
To distribute traffic across microservices and backend systems, modern cloud-native infrastructures primarily rely on load balancers and ingress controllers. Although the main purpose of these components is to increase availability and scalability, they are becoming increasingly important control points in network architecture, making them appealing targets for cyberattacks. Load balancers are frequently viewed as performance-oriented networking components rather than active security enforcement mechanisms, despite their strategic position. To support advanced cybersecurity capabilities, this thesis explores ways to improve load-balancing architecture. The study assesses the efficacy of AI-assisted Web Application Firewalls (WAFs) in identifying real-time threats, investigates the integration of machine learning techniques for proactive vulnerability discovery, and examines protocol-level security issues arising from contemporary transport protocols such as HTTP/3 and QUIC. The study also examines how load-balancing architectures can apply Zero Trust security principles to limit lateral movement in microservice environments and reduce implicit trust. The thesis proposes several defensive improvements for contemporary load balancing systems, drawing on experimental evaluation, architectural design, and security analysis. These include identity-based traffic routing aligned with Zero Trust principles, protocol-aware mitigation techniques for QUIC-specific attacks, and machine-learning-based anomaly detection using load balancer telemetry. The results show that load balancers can be made much more effective as security enforcement points in distributed cloud environments by incorporating AI-driven detection mechanisms and protocol-aware defenses.

Keywords
Load Balancers, Cybersecurity, Machine Learning, Web Application Firewall (WAF), HTTP/3, QUIC, Zero Trust Architecture, Intrusion Detection Systems (IDS), Cloud Security, Microservices Security

1. Introduction

Modern cloud computing environments rely heavily on distributed architectures composed of microservices, container orchestration platforms, and dynamically scalable infrastructure. Recognizing load balancers' critical role in distributing requests to maintain system availability and performance can inspire the audience to appreciate their importance for both performance and security (Katyal & Mishra, 2014; Mishra & Majhi, 2020). As cloud-native systems continue to grow, load balancers have evolved beyond simple traffic distribution mechanisms to include functions like TLS termination, request routing, session persistence, and traffic inspection, highlighting their potential as active security components (Networks, 2023). These components are typically deployed between external networks and backend application servers to optimize performance and manage network traffic efficiently, which underscores their importance in security strategies.

However, the increasing reliance on cloud infrastructures introduces new security challenges, including vulnerabilities in emerging protocols, misconfigured microservices, and sophisticated cyber threats. Although load balancing improves system reliability and resource utilization, the combination of distributed services, dynamic scaling, and high-volume traffic creates a larger attack surface for cyber threats (Ghomia, Rahmani, & Qader, 2017). In large cloud environments, ensuring both efficient traffic distribution and robust security controls has therefore become a significant challenge for infrastructure designers.

While these protocols enhance network efficiency, they also introduce new security concerns, including amplification risks and connection migration abuse, encouraging the audience to consider proactive defenses against these emerging threats and fostering a sense of responsibility.

In parallel, recent advances in machine learning and artificial intelligence have introduced new opportunities for enhancing network security. AI-driven security systems can analyze large volumes of network telemetry data to detect anomalous traffic patterns and identify potential vulnerabilities in real time. These technologies have been increasingly explored in intrusion detection systems, anomaly detection platforms, and intelligent Web Application Firewalls (Katiyar, Nath, Pandey, & Srivastava, 2025). Despite these advancements, relatively limited research has explored how machine learning–based vulnerability discovery, AI-driven security enforcement, protocol-aware defenses, and Zero Trust principles can be integrated directly into load-balancing architectures. As a result, load balancers remain primarily performance-oriented components rather than active security enforcement points (Ghomia, Rahmani, & Qader, 2017). This thesis investigates how modern load balancing architectures can be transformed into active cybersecurity enforcement points that detect vulnerabilities, mitigate protocol-level attacks, and support Zero Trust security models, thereby demonstrating their strategic security role in cloud-native environments.

1.1. Problem Statement

As cloud-native architectures continue to evolve, load balancers have become a central component in the networking stack. However, their growing functionality exposes them to multiple security challenges. First, traditional load balancing systems lack proactive vulnerability detection mechanisms, which should make the audience recognize their critical role and feel the importance of addressing this gap. These issues highlight the need for a more secure and intelligent load-balancing architecture that can detect threats, enforce identity-based trust policies, and mitigate protocol-level vulnerabilities, inspiring confidence in technological advancements.

STARDOM UNIVERSITY

1.2. Research Questions

This thesis explores how innovative security solutions can transform load balancer architectures into proactive defense layers, inviting the audience to contribute to this evolving field.

- **Research Question 1:** How can machine learning techniques be used to proactively identify and analyze vulnerabilities within load balancers and ingress control systems?
- **Research Question 2:** Can AI-based Web Application Firewalls (WAFs) be effectively integrated with load balancers to enhance real-time threat detection and mitigation, addressing practical deployment challenges?
- **Research Question3:** What are the critical protocol-level vulnerabilities introduced by modern standards like HTTP/3 and QUIC, and how can load balancers be strong against them, encouraging awareness of emerging risks?
- **Research Question 4:** How can Zero Trust principles be practically integrated into load-balancing architectures to improve security, and what are the challenges in real-world deployment?
- **Research Question 5:** What best practices and architectural patterns can mitigate security risks specific to Kubernetes ingress controllers and microservice environments?

1.3. Research Objectives

The primary objective of this thesis is to strengthen the cybersecurity of load balancing and ingress systems by designing and implementing innovative tools, techniques, and frameworks. This objective is pursued through the following key goals.

- Investigate AI-based vulnerability discovery tools for load balancers.
- Investigate AI-based WAF for dynamic threat detection and protection in the load balancers.
- Detect and describe stealthy DDoS attacks using pattern analysis and machine learning.
- Propose an improvement to the security status of Kubernetes Ingress controllers.
- Perform a security analysis of HTTP/3 QUIC protocol handling in load balancers and identify possible misconfigurations.
- Design a Zero Trust model for secure communication between load balancers and backend services.

2. Literature Review

This chapter presents a literature review of prior studies and scholarly work on the research topic. It highlights key theories, concepts, and findings that form the foundation of this study as follows:

2.1. Overview

The complexity of securing distributed systems has increased significantly due to the rapid development of contemporary web infrastructures and cloud-native environments. Specifically, load-balancing components are now crucial control points responsible for service availability, traffic routing, and security enforcement. Recognizing their importance can help the audience appreciate the critical role these components play in system security and stability. Vulnerabilities in load-balancing architectures can therefore expose systems to a variety of threats, including lateral movement within microservice environments, distributed denial-of-service attacks, and protocol exploits. Several strategies to reduce these risks have been investigated in prior research, including emerging Zero Trust security models, machine learning-based intrusion detection systems, and conventional rule-based protection mechanisms. To determine the current state of knowledge and pinpoint the constraints that drive the methodology proposed in this thesis, this section emphasizes the need for AI-driven security frameworks integrated with load balancers to effectively mitigate protocol vulnerabilities.

2.2. Machine Learning for Attack Detection

An intrusion detection system based on machine learning techniques was proposed in a study by (Katiyar, Nath, Pandey, & Srivastava, 2025) to enhance the detection of cyberattacks in network environments. The authors classified malicious activity and examined network traffic patterns using several machine learning algorithms. Highlighting the potential of machine learning can inspire the audience to see these methods as promising solutions for improving security. However, the study does not address how machine learning can be integrated into load-balancing architectures or handle protocol-specific vulnerabilities, revealing a gap in applying AI directly to load balancer security.

A study on intrusion detection systems that use machine learning techniques to enhance attack detection in network environments was presented (Chentoufi & Chougdali, 2021). In addition to discussing various IDS types, such as host-based and network-based systems, the paper describes how supervised and unsupervised machine learning algorithms are used to identify malicious activity. To reduce data dimensionality and enhance detection performance, the authors proposed a method that combines Principal Component Analysis (PCA) with classification algorithms. According to their findings, combining dimensionality reduction with machine learning classifiers can improve detection accuracy while lowering computational complexity. However, the study does not particularly address security mechanisms within contemporary load balancing architectures or protocol-level attack mitigation; instead, it primarily focuses on general network intrusion detection.

2.3. Load Balancer and Protocol-Level Security

A thorough investigation of security flaws in the QUIC protocol—the transport layer that HTTP/3 uses—was carried out by (Chatzoglou, Kouliaridis, Karopoulos, & Kambourakis, 2022). The study offers a thorough analysis of prior research on QUIC security and categorizes attacks against the protocol into categories such as cryptographic, handshake, fuzzing,

transport-layer, and privacy attacks. The authors conducted practical security testing on several QUIC server implementations using fuzzing techniques in addition to the literature review. Numerous flaws that could lead to resource depletion or denial-of-service attacks against servers were identified in the experiments. These results show that while QUIC's built-in encryption enhances web security and performance, its implementations are still relatively new and may still have security flaws that hackers could exploit. This highlights a gap in understanding how protocol vulnerabilities translate into real-world security risks for load balancers and microservices, warranting further investigation.

Using a publicly accessible dataset, Chatzoglou et al. (2023) performed an empirical security analysis of the HTTP/3 protocol by comparing its behavior with that of HTTP/2. The study sought to determine whether, despite the protocol's performance gains, switching to HTTP/3 introduces new security threats. The authors examined HTTP traffic patterns and assessed possible weaknesses in network behavior and protocol implementation. Emphasizing the need for continuous security evaluation can encourage the audience to adopt a proactive approach to emerging web protocols. According to their findings, even though HTTP/3 benefits from the security features of the QUIC transport protocol, attackers can still exploit several implementation-level flaws and unusual traffic patterns. To ensure the safe implementation of new web protocols in contemporary network infrastructures, the study emphasizes the importance of ongoing security assessment.

(Gbur & Tschorsch, 2023) looked into QUICforge, a security flaw in the QUIC protocol that makes client-side request forgery attacks possible. The study shows how attackers can manipulate client requests and access unauthorized network resources by exploiting QUIC-based communication. The authors demonstrate through experimental analysis that the vulnerability stems from the way QUIC connections are established and managed, thereby enabling malevolent actors to circumvent certain security limitations. Their results point to potential risks associated with the growing use of QUIC- and HTTP/3-based services. To prevent QUIC connections from being abused in contemporary web infrastructures, the study highlights the need for stronger security measures and validation procedures.

2.4. Security in Load Balancer and Microservice Architectures

In their investigation of security issues in microservice architectures, Khare et al. (2024) proposed a framework that combines load balancing with Role-Based Access Control (RBAC) techniques. The study emphasizes that inadequate access control and communication among distributed services frequently pose security threats to microservices environments. To solve these problems, the authors used load balancing to distribute incoming traffic and enforced RBAC policies to regulate access for users and services. Their strategy seeks to prevent unwanted access to microservices while improving system availability. The findings show that in distributed application environments, load balancing combined with structured access control mechanisms can enhance security and performance.

In order to handle machine learning inference requests on serverless computing platforms while upholding Service Level Agreements (SLAs), (Mahmoudi & Khazaei, 2022) proposed MLProxy, a reverse proxy system. The study highlights that effective traffic routing is necessary in serverless architectures for machine learning applications to guarantee low latency, high availability, and resource optimization. MLProxy enhances system performance and reliability by intelligently scheduling and balancing incoming requests across available computing resources. The paper illustrates the significance of load balancing mechanisms in contemporary distributed and microservice-based ML systems, which is pertinent for creating

safe and robust load balancing frameworks, even though its main focus is SLA-aware traffic management.

2.5. AI-Based Traffic Management and Load Balancing

In order to enhance cloud service delivery, " (Moharir, Kuppuraju, & Kumar, 2025) looked into advanced load balancing techniques that make use of artificial intelligence. The study emphasizes that conventional load balancing techniques often struggle to manage dynamic traffic patterns and rapidly shifting workloads in contemporary cloud environments. The authors suggested AI-based methods to evaluate system behavior and automatically distribute incoming requests across available resources to solve this problem. Their strategy seeks to increase overall service reliability, lower latency, and improve resource utilization. The findings show that, compared with conventional static load balancing techniques, AI-driven load balancing strategies can significantly improve cloud performance and enable more flexible traffic management.

To address the challenges of dynamic cloud environments, Chawla (2024) proposed an adaptive load-balancing strategy based on reinforcement learning. The study emphasizes that conventional load balancing algorithms often struggle to adapt effectively to changing workloads and resource availability in distributed cloud systems. By continuously learning from system conditions, the proposed reinforcement learning model adjusts traffic distribution strategies. Better system performance and more effective resource use are made possible by this adaptable strategy. The findings show that dynamically optimizing workload distribution in real time can outperform conventional static methods.

2.6. Zero Trust and Modern Network Security Architectures

(Rajendran, Rai, Anumula, & Agrawal, 2025) investigated the use of identity federation mechanisms in microservices architectures to implement a Zero Trust security model. The study highlights that for contemporary distributed systems where services interact dynamically across cloud environments, conventional perimeter-based security techniques are inadequate. The authors proposed a security framework that ensures ongoing authentication and authorization between microservices by verifying user and service identities before granting access to resources. To provide secure communication and centralized identity management across services, their method incorporates identity federation techniques. The findings show that by preventing unauthorized access and reducing the likelihood of lateral movement in microservice environments, implementing Zero Trust principles can greatly enhance security.

2.7 Research Gap

Even though intrusion detection systems, AI-based traffic management, and contemporary security architectures have advanced significantly, current research often addresses these topics in isolation. While some studies address load-balancing optimization or protocol-level vulnerabilities independently, many focus on enhancing attack detection using machine learning. However, integrated solutions that can concurrently address security, performance, and protocol-level threats are necessary for contemporary cloud-native infrastructures. Specifically, the growing use of QUIC and HTTP/3 creates new attack surfaces that are beyond the capabilities of conventional load balancing and security measures. To mitigate emerging protocol-level threats in distributed environments, a comprehensive architecture that integrates secure load-balancing mechanisms with intelligent attack detection is required.

The reviewed studies address different aspects of cybersecurity, traffic management, and modern cloud architectures. Some research focuses on machine learning-based intrusion detection systems, while others investigate protocol-level vulnerabilities in emerging web technologies such as HTTP/3 and QUIC. Additionally, several studies explore intelligent load balancing strategies and Zero Trust security models for distributed systems.

3. Methodology

This chapter explains the study's methodology. It details adopting a methodological approach, applying machine learning for vulnerability discovery, evaluating AI-based web application firewalls, analyzing QUIC and HTTP/3 vulnerabilities, integrating zero trust, and securing Kubernetes Ingress. The chapter concludes with a summary as follows:

3.1. Methodological Approach

This research adopts a multi-layer experimental and architectural methodology, emphasizing the importance of combining machine-learning experimentation, Web Application Firewall evaluation, protocol-level security analysis, and secure architecture design to build a comprehensive understanding of cybersecurity challenges in modern load-balancing infrastructures.

The methodology was designed to evaluate security mechanisms across multiple layers of the load-balancing ecosystem, including behavioral traffic analysis, application-layer protection, transport-protocol vulnerabilities, and cloud-native architecture security.

The research process consisted of five main methodological stages:

- Machine learning–based traffic analysis for vulnerability discovery.
- Evaluation of AI-based Web Application Firewall effectiveness.
- Protocol-level vulnerability analysis in QUIC and HTTP/3.
- Zero Trust architectural integration within load balancers.
- Security analysis and mitigation strategies for Kubernetes ingress architectures.

Each stage addresses one of the research questions while contributing to a unified defensive framework.

3.2. Methodology for Machine Learning–Based Vulnerability Discovery

How can machine learning techniques be used to proactively identify and analyze vulnerabilities within load balancers and ingress control systems?

To answer this question, an experimental machine learning framework was developed to analyze behavioral patterns in load-balanced traffic and detect anomalous activity that may indicate security vulnerabilities.

3.2.1. Experimental Environment

A containerized microservice environment was deployed using Kubernetes, with HAProxy acting as the primary load balancer. Traffic monitoring and system telemetry were collected using Prometheus, which provided time-series metrics describing network behavior and load balancer performance.

This environment enabled the simulation of realistic traffic patterns, including both legitimate user requests and malicious activity, to ensure the study's findings apply to real-world scenarios.

3.2.2. Dataset and Feature Extraction

Network telemetry data was collected from the load-balancing layer and transformed into behavioral features. These features included:

- Request rates per source IP
- Connection durations
- Traffic burst patterns.
- Load balancer routing statistics
- Connection failure rates

Unlike traditional intrusion detection systems that rely on packet payload inspection, this study focuses on behavioral telemetry extracted directly from load balancing infrastructure.

The dataset was divided into training and testing subsets using an 80:20 ratio while maintaining class balance through stratified sampling.

3.2.3. Machine Learning Models

Labeled, unlabeled, or hybrid datasets serve as the foundation for training machine learning algorithms. Accordingly, the choice of a machine learning model is largely determined by the type of data required for the specific task. In this study, four categories of machine learning approaches were applied to labelled data. Subsequently, several machine learning models were systematically evaluated to assess their effectiveness in detecting abnormal traffic behavior, including:

- Random Forest Machine Learning
- Support Vector Machine Learning
- Logistic Regression Machine Learning
- Decision Trees Machine Learning

The models were evaluated using a confusion matrix, which includes accuracy, precision, recall, and F1-score to assess their ability to detect attack patterns from load-balancer-level telemetry alone.

3.3. Methodology for AI-Based Web Application Firewall Evaluation

Can AI-based Web Application Firewalls enhance the detection and mitigation of real-time threats at the load balancing layer?

To address this research question, a comparative security experiment was conducted to evaluate the detection capabilities of a traditional Web Application Firewall and a machine learning–based detection system. This section aims to demonstrate how advanced detection systems can significantly enhance real-time threat mitigation at the load-balancing layer, thereby fostering confidence in innovative security approaches.

3.3.1. Experimental Setup

The experiment used the ModSecurity WAF, integrated with NGINX, as the traditional rule-based defense system. Attack traffic and benign requests were generated and processed through the load-balancing environment.

Simultaneously, a machine-learning detection model analyzed traffic patterns using behavioral features extracted from load balancer telemetry.

3.3.2. Attack Scenarios

The experiment included multiple simulated attack categories, including:

- SQL injection attempts
- Cross-site scripting attacks
- HTTP flooding and denial-of-service attempts
- Normal user traffic

3.3.3. Comparative Evaluation

The detection results of the rule-based WAF and the machine learning model were compared using a unified dataset. Each request was classified as:

- True Positive (TP)
- True Negative (TN)
- False Positive (FP)
- False Negative (FN)

This comparison enabled the evaluation of whether AI-based detection can complement traditional rule-based Web Application Firewalls, particularly in detecting previously unseen attack patterns.

3.4. Methodology for QUIC and HTTP/3 Vulnerability Analysis

What are the critical protocol-level vulnerabilities introduced by modern standards like HTTP/3 and QUIC, and how can load balancers be strengthened against them?

To answer this question, a protocol-level security analysis was conducted, focusing on the behavior of modern transport protocols used by web infrastructure. This section underscores the thoroughness of the research in identifying critical security risks posed by modern standards such as HTTP/3 and QUIC, reassuring the audience of the study's depth and reliability.

3.4.1. Protocol Behavior Analysis

The study examined key QUIC and HTTP/3 design features that influence load balancing behavior, including:

- Connection ID–based routing
- Stateless retry validation
- 0-RTT early data
- Connection migration
- UDP transport amplification risks

3.4.2. Threat Modeling

Several attack scenarios were modeled to evaluate how these protocol behaviors could be exploited against load-balancing systems, including:

- UDP reflection and amplification attacks
- Handshake flooding attacks
- Connection migration exploitation
- Replay attacks using 0-RTT data
- Connection ID entropy manipulation

3.4.3. Defensive Architecture Development

Based on the identified vulnerabilities, an AI-augmented defensive architecture was designed to mitigate protocol-level threats through behavioral monitoring and adaptive mitigation mechanisms.

3.5. Methodology for Zero Trust Integration

How can Zero Trust principles be integrated into load-balancing architecture?

Highlighting this integration aims to build trust in the security approach, reassuring cybersecurity researchers and network engineers of its reliability.

3.5.1. Identity-Centric Routing Model

Traditional load-balancing architectures rely primarily on network-level attributes such as IP addresses and ports. This research proposes shifting to identity-based routing mechanisms that incorporate verified workload identities and behavioral trust scores.

3.5.2. Zero Trust Enforcement Mechanisms

The architecture introduces several Zero Trust enforcement mechanisms at the load balancing layer, including:

- Strong service authentication
- Continuous behavioral verification
- Policy-based routing decisions
- Microservice identity validation

This approach transforms the load balancer from a passive routing component into an active security enforcement gateway.

3.6. Methodology for Kubernetes Ingress Security

What best practices and architectural patterns can mitigate security risks specific to Kubernetes ingress controllers and microservice environments?

To address this research question, a security architecture analysis was conducted, focusing on Kubernetes ingress controllers and microservice networking.

3.6.1. Security Risk Identification

Common security risks within Kubernetes ingress deployments were analyzed, including:

- Unintended service exposure
- Traffic interception risks
- Privilege escalation in ingress controllers
- Lateral movement between microservices
- Weak service authentication

3.6.2. Secure Architecture Design

A layered defensive architecture was proposed to mitigate these risks using security controls such as:

- Network policies for service isolation
- Mutual TLS authentication between services
- Secure ingress routing policies
- Role-based access control for ingress controllers
- Service mesh–based identity verification

These architectural patterns provide practical guidelines for securing microservice environments operating behind load-balancing infrastructure.

4. Results

Based on the data gathered and examined using the chosen research methodology, the study's conclusions are presented in this chapter. The five questions listed in Chapter one and the study objectives are used to organize the results.

The experiments and analysis carried out are presented in this section. The goal is to assess how well the proposed architecture addresses the security issues in contemporary load-balancer environments. Analyzing system behavior under various attack scenarios and evaluating the effectiveness of the mitigation mechanisms implemented are the main goals of the experiments.

The research questions and comprehensive answers are presented in the ensuing subsections. To show how the suggested method enhances the load-balancing infrastructure's security, detection capacity, and resilience, each question is examined using quantitative findings and a system-level assessment. Each question is answered individually in a separate subsection in the following order:

1- How can machine learning techniques be used to proactively identify and analyze vulnerabilities within load balancers and ingress control systems?
2- Can AI-based Web Application Firewalls enhance the detection and mitigation of real-time threats at the load balancing layer?
3- What are the critical protocol-level vulnerabilities introduced by modern standards like HTTP/3 and QUIC, and how can load balancers be strong against them?
4- How can Zero Trust principles be integrated into load-balancing architecture?
5- What best practices and architectural patterns can mitigate security risks specific to Kubernetes ingress controllers and microservice environments?

4.1. Machine Learning–Driven Vulnerability Detection in Load Balancing Environments

This section will answer the question “How can machine-learning techniques be used to proactively identify and analyze vulnerabilities within load balancers and ingress control systems?”

Cybersecurity might go through a significant transformation due to AI and machine learning. To detect stealthy attacks before rule-based engines, Microsoft researchers showed that deep learning techniques can analyze behavioral patterns in massive volumes of telemetry data (Microsoft Defender Security Research Team, 2020). Similarly, the IEEE has demonstrated that AI can support adaptive threat response in dynamic infrastructure environments, such as edge proxies and microservice meshes, and can also improve detection (IEEE-USA, 2025).

While machine learning-based intrusion detection systems (IDS) have demonstrated promising results in identifying abnormal traffic patterns, traditional security mechanisms such as rule-based firewalls and signature-based IDS often fail to detect new or evolving attack patterns.

(Katiyar, Nath, Pandey, & Srivastava, 2025) assessed several conventional machine learning classifiers using the KDD intrusion detection dataset. To train the models on a large portion of the data and reserve a smaller portion for performance evaluation, their study split the dataset into training and testing subsets using an 80:20 ratio. Their study's findings showed that, when trained on structured network data, traditional machine learning methods can produce excellent detection performance. Unlike earlier studies, the current study does not use packet payload inspection. Instead, it aims to achieve a comparable detection capability using only flow-level load balancer metrics.

4.1.1. Experiment Structure

This section will show the experimental environment of 3 virtual machines as follows:

1- Ubuntu Application VM that contains an ingress control system and hosts 2 simple hello apps.
2- Ubuntu LB VM hosting a load balancer integrated with a logging and monitoring service.
3- Kali Linux Attacker VM

Application VM

The Application VM hosts Minikube hello application containers and Nginx to forward requests to the hosted.

Load Balancer VM

The LB VM acts as a gate, forwarding requests to the applications via a round-robin load-balancing algorithm, and is integrated with Prometheus to track traffic.

Attacker VM

The attacker VM will act as a traffic generator, attacking the LB VM with various attack scenarios and methods.

4.1.2. Experiment Technologies

This section will explain the technology stack used to process the experiment and why each was chosen.

Ingress Control System

Because Kubernetes is a popular open-source platform for automating the deployment, scaling, and management of containerized applications, it was chosen. In contemporary cloud-native environments, where load balancers and ingress controllers control access to external services, Kubernetes is frequently used (Kubernetes, 2025).

Load Balancer

Because HAProxy is a popular open-source load balancer and reverse proxy with excellent performance and dependability, it was chosen. Additionally, HAProxy has built-in support for exporting operational metrics, which can be used as input features for machine learning models (HAProxy, n.d.).

Logging and Monitoring Service

Prometheus, an open-source monitoring and alerting system for gathering time-series metrics from infrastructure components, was chosen. Prometheus can effectively collect system metrics and store them as time-series data for machine-learning analysis (Prometheus, n.d.).

Machine Learning Model

A Random Forest classifier was selected for its interpretability, scalability, robustness, and ability to handle high-dimensional structured data, making it well-suited for detecting complex attack patterns (Nawaz, Tahira, Shah, Ali, & Tahir, 2025).

4.1.3. Datasets

A bunch of normal and several abnormal traffic categories were generated by bash scripts using Kali Linux tools from the attacker to the load balancer during the period between 2025-10-11T20:00:00 and 2025-10-11T22:00:00, as shown in Table 1. This data will be saved as a CSV file for model training.

Table 1: Datasets

Scenario	Start Time	End Time
Normal	2025-10-11T20:19:03	2025-10-11T20:24:07
DoS (hey)	2025-10-11T20:41:21	2025-10-11T20:41:42
DoS (ab)	2025-10-11T21:04:52	2025-10-11T21:05:42
Forgery	2025-10-11T21:09:06	2025-10-11T21:16:28
Probe (endpoints)	2025-10-11T21:20:03	2025-10-11T21:20:06
Probe (port scanning)	2025-10-11T21:22:26	2025-10-11T21:22:26

4.1.4. Data Collection and Evaluation

A Python script used a simple Random Forest classifier, trained on aggregated Prometheus data. The data collection pipeline queried HAProxy performance indicators (requests, bytes, connection errors) because they are standard indicators of LB health in the monitoring through the Prometheus API, according to (Mouzakitis, 2015) as follows:

- haproxy_frontend_http_requests_totals
- haproxy_backend_http_requests_total
- haproxy_backend_connection_errors_total

- haproxy_backend_client_aborts_total
- haproxy_frontend_bytes_in_total
- haproxy_frontend_bytes_out_total

The metrics were collected over the experiment window with a 5-second sampling step and a 10-second aggregation window. Each window was labeled based on the corresponding traffic scenario defined in Table 4.2.1. The dataset was split into training and testing subsets (80:20 ratio) using stratified sampling to maintain consistent class distribution, thereby improving model reliability and evaluation accuracy, similar to (Katiyar, Nath, Pandey, & Srivastava, 2025).

(Kumar, 2014) suggests the confusion matrix as an evaluation method for intrusion detection systems (IDS).

4.1.5. Experiment Result

After training, the model achieved the following performance:

- Training accuracy: 1.00
- Testing accuracy: 0.98

The model achieved the following metrics:

- Accuracy of 97.83%.
- Precision of 68.00%.
- Recall of 79.51%.
- F1-score of 72.03%.

The high accuracy demonstrates successful classification between benign and malicious traffic, which should reassure the audience about the potential of these methods, even as the relatively low precision highlights areas for further improvement.

4.1.6. Summary

While the experiment used a simple modeling mechanism and simple traffic datasets, the results indicate that the model successfully learns to distinguish normal from abnormal traffic, demonstrating strong generalization and effective learning of traffic behavior patterns. The results encourage more enhancements. The ML model must be improved and trained over more real-life scenarios and a large amount of datasets.

The results show that machine learning techniques can be an effective part of load balancers' security, regardless of the simplicity of the ML model or the datasets used. It can be used to take immediate action and log a report when it detects abnormal traffic. The use of Prometheus as a real-time data source allowed continuous feature extraction and automated labeling, making this approach adaptable to live monitoring environments.

(Sivaraman, 2024) presented a novel machine-learning-based model designed for threat detection and monitoring used in Kubernetes container networks. The model enhanced the principles of DevSecOps (integrating security into DevOps). However, the proposed model improves both security and resilience in Kubernetes environments; the author also noted some limitations in the dataset's quality. The increase in container system usage necessitates further investigation into developing and producing additional models for threat detection and prevention.

4.2. Empirical Analysis of ML-Based Threat Detection Versus WAF Rule-Based Decisions

This section will answer the question "Can AI-based Web Application Firewalls enhance the detection and mitigation of real-time threats at the load balancing layer?"

Modern load balancers such as NGINX, HAProxy, and F5 play a crucial role in ensuring application performance, availability, and scalability. In addition to managing sophisticated functions such as SSL/TLS termination, health monitoring, traffic routing, and layer 7 request processing, they are primarily used to distribute incoming traffic across multiple backend servers. Furthermore, load balancers are valuable targets for adversaries looking to take advantage of weaknesses at the network and application layers because they are essential control points in enterprise security architectures (Networks, What Is a Load Balancer?, n.d.) (Networks, Load Balancing 101: Nuts and Bolts, n.d.).

Web Application Firewalls (WAFs) are tools that filter requests and responses between the client and the web application, detecting unauthorized access attempts, such as SQL injection and cross-site scripting (XSS). ModSecurity is an open-source WAF used to protect applications against common vulnerabilities using a generic attack detection rule called OWASP CRS (MK, Bala, Sonti, & KP, 2025).

The research (Otero-Mosquera, Lo´pez-Bravo, Tub´ıo-Figueira, & Iglesia, 2025) proposed an AI intrusion detection system on Shadow Daemon WAF with datasets with various payloads, like legal, XSS, SQL injection, and shell injection, as shown in Figure 1. The authors used two multiclass classification models, based on SVM and MLP neural networks, respectively.

TABLE 1: Dataset1: number of payloads of each category.

Type payload	Number
Legal	100,496
Cross-site scripting	17,486
SQL injection	35,317
Shell command injection	2566

TABLE 2: Dataset2: number of payloads of each category.

Type payload	Number
Legal	358
Cross-site scripting	191
SQL injection	110
Shell command injection	330
Cross-site request forgery	73
Server-side request forgery	73
XML external entities	65

TABLE 8: Metrics by classifier for the Dataset1 scenario.

	Shadow Daemon	SVM-ML Shadow Daemon	MLP-ML Shadow Daemon
Accuracy	0.9368	0.9978	0.9963
Precision	0.9582	0.9978	0.9938
Recall	0.8608	0.9960	0.9958
F1-score	0.9069	0.9969	0.9948
FPR	0.0209	0.0012	0.0034
FNR	0.1392	0.0040	0.0042

TABLE 10: Metrics by classifier for the Dataset2 scenario.

	Shadow Daemon	SVM-ML Shadow Daemon	MLP-ML Shadow Daemon
Accuracy	0.6775	0.7367	0.7890
Precision	0.8628	0.8321	0.8639
Recall	0.6425	0.7827	0.8300
F1-score	0.7366	0.8066	0.8466
FPR	0.2402	0.3715	0.3073
FNR	0.3575	0.2173	0.1700

Figure 1: Datasets and results in the (Otero-Mosquera, Lo´pez-Bravo, Tub´ıo-Figueira, & Iglesia, 2025) **study**

This section compares the detection performance of a raw ModSecurity WAF installed on an NGINX load balancer with that of a ModSecurity WAF integrated with a simple ML detection model.

4.2.1. Experiment Structure

This section will show the experimental environment of 3 virtual machines as follows:

1- Ubuntu LB with ModSecurity WAF VM.
2- Ubuntu LB with ModSecurity ML-WAF VM.
3- Kali Linux Attacker VM

Load Balancer with WAF VM

A Node that contains a vulnerable application and a load balancer with WAF.

Load Balancer with ML-WAF VM

A Node that contains a vulnerable application and a load balancer with WAF with an ML scoring model.

Attacker VM

The attacker VM will act as a traffic generator to attack both LB VMs using various attack scenarios and methods.

4.2.2. Experiment Technologies

This section will explain the technology stack used to process the experiment and why each was chosen.

Load Balancer

NGINX was selected as the load balancer because it is open-source software, widely used in modern web infrastructures, and capable of handling thousands of concurrent connections. NGINX also supports integration with modules such as ModSecurity, enabling WAF protection directly at the edge of the application infrastructure (NGINX, n.d.).

WAF

ModSecurity was selected as the Web Application Firewall, using the OWASP Core Rule Set, which provides a standardized collection of security rules. ModSecurity can integrate with web servers such as NGINX and is open-source software (ModSecurity, n.d.).

Machine Learning Model

A simple Logistic Regression was used as an ML model for ModSecurity because it combines features linearly with learned weights to predict a binary outcome, making it suitable for intrusion detection, such as WAF operations, and offering fast training and low computational overhead (Chentoufi & Chougdali, 2021).

Datasets

A total of 100 normal and 400 abnormal traffic samples from several categories were generated by the attacker using bash scripts and Kali Linux tools for both load balancers, as shown in Table 2. Each request includes a unique flag in the header to be distinguished from other requests. The dataset contains a DoS network-related abnormal threat, which was not in (Otero-Mosquera, Lo´pez-Bravo, Tub´ıo-Figueira, & Iglesia, 2025).

Table 2: Traffic Types

Type	Count
Normal	100
SQL injection	100
XSS	100
EDGE	100
DoS	100

4.2.3. Data Collection and Evaluation

For the regular WAF, the ModSecurity logs will be used to extract the data. The ML-WAF will work with a simple Python ML scorer, using an API triggered by a Python tailer process that monitors the ModSecurity logs for all requests and makes a decision. The Python ML scorer uses a Logistic Regression classifier as the ML model, multiplying request features such as SQL indicators and XSS tokens by learned weights stored in a model file, and then comparing the resulting score against a classification threshold to produce decisions.

Similar to the previous Machine Learning–Driven Vulnerability Detection in Load Balancing Environments experiment, Kumar (2014) suggests using the confusion matrix to evaluate intrusion detection systems (IDS).

4.2.4. Experiment Results

Three experimental runs were conducted due to implementation errors.

Experiment Result with Wrong Labeling Format

The first experimental run revealed a misalignment between the raw ModSecurity WAF logs and the ML decision logs. The issue resulted from the X-Test-ID tagging mechanism being injected into every request, as it was designed to generate IDs from timestamps, which are not identical across both VMs. This issue resulted in several comparison results greater than the 500 sent requests, as shown in Table 3.

Table 3: Wrong Labeling Format Result

Metric	Value
Total Samples	1303
True Positive (TP)	991
False Positive (FP)	312
False Negative (FN)	0
True Negative (TN)	0
Accuracy	0.761
Precision	0.761
Recall	1
F1-score	0.864

Explain that fixing the X-Test-ID tagging mechanism to use a unique, identical tag for each request was essential for accurate cross-VM comparison, ensuring reliable results.

Experiment Result with Misclassification of Normal as DoS

The second experimental run shows that the ML decision was misclassifying a normal request as a DoS activity. The reason for this issue is that the ML scorer is caching the request header to check the rates of incoming requests. Since all requests come from a single attacking machine, the model will always classify these requests as DoS activity because high-frequency traffic patterns resemble attack bursts within that narrow window, as shown in Table 4.

Table 4: Misclassification of Normal as DoS Result

Metric	Value
Total Samples	500
True Positive (TP)	300
False Positive (FP)	200
False Negative (FN)	0
True Negative (TN)	0
Accuracy	0.6
Precision	0.6
Recall	1
F1-score	0.75

This issue was fixed by increasing the time between normal requests and increasing the frame-time duration in the ML model to prevent normal high-rate requests from being misclassified as DoS activities.

Final Experiment Result

After fixing the previous issues, the final results show an excellent confusion matrix for analysis, as shown in Table 5.

Table 5: Final Result

Metric	Value
Total Samples	500
True Positive (TP)	300
False Positive (FP)	100
False Negative (FN)	0
True Negative (TN)	100
Accuracy	0.8
Precision	0.75
Recall	1
F1-score	0.857

4.2.5 Summary

4.3. Protocol-level vulnerability mitigations in QUIC-based HTTP/3 by the load balancer architecture

This section will answer the question "What are the critical protocol-level vulnerabilities introduced by modern standards like HTTP/3 and QUIC, and how can load balancers be strong against them?"

The Internet Engineering Task Force (IETF) produced a transition from TCP-based HTTP/2 to QUIC-based HTTP/3, representing a new structural design of the internet transport standards. Quick UDP Internet Connections (QUIC) operates over UDP rather than TCP's connection semantics and integrates TLS 1.3 directly at the transport layer to encrypt most metadata. The QUIC protocol also enables a feature called 0-RTT, which allows clients to send application data before the handshake completes (J. Iyengar & M. Thomson, 2021) (Ed & S. Turner, 2021) (M. Bishop, 2022). While this design reduces latency and head-of-line blocking, it increases protocol-level complexity and reduces middle-box visibility.

(Chatzoglou, Kouliaridis, Karopoulos, & Kambourakis, 2022) researched the security of HTTP/3 through comparative experimentation with HTTP/2, verifying that legacy attack patterns from HTTP can be partially transposed into QUIC by exhibiting new amplification characteristics.

(Chatzoglou, Kouliaridis, Kambourakis, Karopoulos, & Gritzalis, 2023) performed comprehensive evaluations of QUIC security on six QUIC-enabled production servers. The assessment identified several zero-day vulnerabilities using fuzzing and logical state testing. The assessment showed resource exhaustion, protocol-state violations, and erroneous enforcement of amplification.

QUIC-based HTTP/3 is threatened by protocol-level exploitation. The QUIC-forge paper (Gbur & Tschorsch, 2023) revealed a new client-side forgery mechanism inherent in the QUIC packet-processing model: it amplifies UDP traffic, bypasses anti-amplification logic, and masquerades as other protocols. This directly verifies that theoretical safeguards in the QUIC RFC do not imply secure real-world mechanisms.

Cross-layer interactions between QUIC, encrypted DNS (DoH/DoQ), and HTTP/3 introduce new traffic behavior patterns, as shown by (Sengupta, Kosek, Fries, Ferlin-Reiter, & Bajpai, 2024). These exchanges have the potential to evade detection and facilitate covert misuse throughout encrypted stacks. (Sengupta, Kosek, Fries, Ferlin-Reiter, & Bajpai, 2024) demonstrates that QUIC security needs to be examined in the context of ecosystem interactions rather than assessed in isolation.

These works show that QUIC's security deployment remains the prerogative of implementation. Also, load balancers, which are based on TCP semantics and passive inspection, make them exposed to a new class of attacks that legacy controls do not mitigate. This section aims to resolve the issue by:

1- Identify protocol-level vulnerabilities shown in recent literature.
2- Allocate them to architectural weaknesses in load balancers.
3- Propose a mitigation mechanism.

4.3.1. protocol-level vulnerabilities

The protocol-level vulnerabilities will be identified and explained in the following sections.

UDP Amplification and Reflection

Since QUIC inherits UDP's properties, UDP's spoofability also applies to QUIC. Even though the implementation of RFC 9000 enforces a 3x amplification limit before address validation)J. Iyengar و M. Thomson(2021 ،, Gbur & Tschorsch (2023) show inconsistent implementation.

Even though QUIC has address validation and anti-amplification features, real-world implementations show inconsistent enforcement, especially in multi-layer architectures with load balancers. UDP-based amplification attacks are enabled when QUIC traffic is routed without early address validation or amplification accounting, turning the load balancer from a mitigation layer into a reflection facilitator (Buchet & Pelsser, 2025).

Based on these observed deployment weaknesses, the following mitigations help at the load-balancer level:

1- Enforce the QUIC anti-amplification limits (3× rule) at the load balancer before sending traffic to the backend servers.
2- Verify client IP addresses before allocating resources and make Stateless Retry mandatory at the load balancer.
3- Stop spoofing and replaying among dispersed nodes, cryptographically bind address validation tokens to source IP, timestamp, and load balancer instance secrets.
4- Counteract high-rate spoofing attempts at reflection, implement per-source rate limiting for QUIC Initial packets.
5- Install QUIC-aware load balancing that can separate established 1-RTT sessions from handshake traffic and parse initial packets.

0-RTT Replay Abuse

The Zero Round-Trip Time (0-RTT) is an early data performance supported by QUIC. However, 0-RTT is replayable and does not guarantee forward secrecy (Ed & S. Turner, 2021).

Distributed load balancers lack synchronized replay caches and may accept duplicate early data on their own. This issue might be resolved as follows:

1- Install a cluster-wide replay cache.
2- Limit 0-RTT to HTTP methods that are idempotent.
3- Implement anomaly scoring for repeated data patterns.

Connection Migration Exploitation

QUIC employs Connection IDs (CIDs) in place of the 5-tuple (source and destination IP/port along with the protocol) for distinguishing connection identities, enabling a connection to alter its underlying 5-tuple without needing a new handshake for establishment, a process known as connection migration (Grübl, Niu, Assen, & Stiller, 2025). Under the migration of connections, attackers might be able to manipulate CIDs or state tables. This issue could be resolved by:

1- Use CID mapping to determine the route.
2- Implement CID entropy checks.
3- Employ anomaly detection for this type of network activity.

State Exhaustion and Memory Abuse

According to Chatzoglou et al. (2023), fuzzing campaigns show that the misuse of streams and incorrect transport parameters can inflate per-connection state. load balancers are state-exhaustion targets before backend intervention, and this issue is resolved by:

1- Lazy allocation of states.
2- IP-specific memory limits.
3- Progressive handshake validation.

Encrypted Transport and Loss of Observability

As stated by Ed & S. Turner (2021), load balancers cannot examine sequence numbers, flags, and flow behavior because QUIC encrypts transport metadata, eliminating passive TCP inspection. This issue could be resolved by:

1- Extract encrypted features (packet size, timing, CID rotation).
2- Use anomaly detection instead of signature inspection.
3- Integrate AI-based behavioral modeling.

4.3.2. Architectural Implications for Load Balancers

The load balancer must evolve from a straightforward traffic distributor into a QUIC-aware security enforcement layer to safely handle QUIC-based traffic. Five fundamental design requirements can be used to summarize the architectural implications:

1- Enforce Anti-Amplification and Address Validation by:
- Enforce the 3× anti-amplification rule before sending traffic to backend servers.
- Confirm client IP ownership by making Stateless Retry mandatory.
- Cryptographically link validation tokens to source IP, timestamp, and LB secret.
- Apply per-source rate limiting for QUIC Initial packets.
- Separate established 1-RTT sessions from handshake traffic.

This guarantees that traffic based on reflection or spoofing is filtered at the edge.
2- Secure 0-RTT Handling by:
- Install a replay cache for the entire cluster.
- Allow only idempotent HTTP methods to use 0-RTT.
- Use anomaly scoring to find recurring early-data patterns.

This prevents replay attacks across distributed load balancer nodes.
3- Control Connection Migration by:
- Use deterministic CID mapping for backend routing.
- Verify the CID entropy.
- Identify the frequency of abnormal migration.

This prevents exploitation of routing and CID manipulation.
4- Prevent State Exhaustion by:
- Implement lazy state allocation.
- Apply IP-specific memory limits.
- Use progressive handshake validation before significant cryptographic allocation.

This lessens the likelihood of handshake flooding and memory abuse.
5- Maintain Observability Under Encryption by relying on the following:
- Packet size and timing patterns.
- Behavior of CID rotation.

- Handshake progression signals

Behavioral analysis must replace payload inspection in security decisions.

4.3.3. AI-Augmented Defensive Architecture

Transforming load balancers from passive distribution components into active ones is needed due to the limitations imposed by encryption and high-speed UDP transport, for which there are insufficient solutions. An AI-augmented architecture will provide adaptive and behavior-driven protection for all identified classes. The proposed AI-augmented defensive architecture is illustrated in Figure 3.

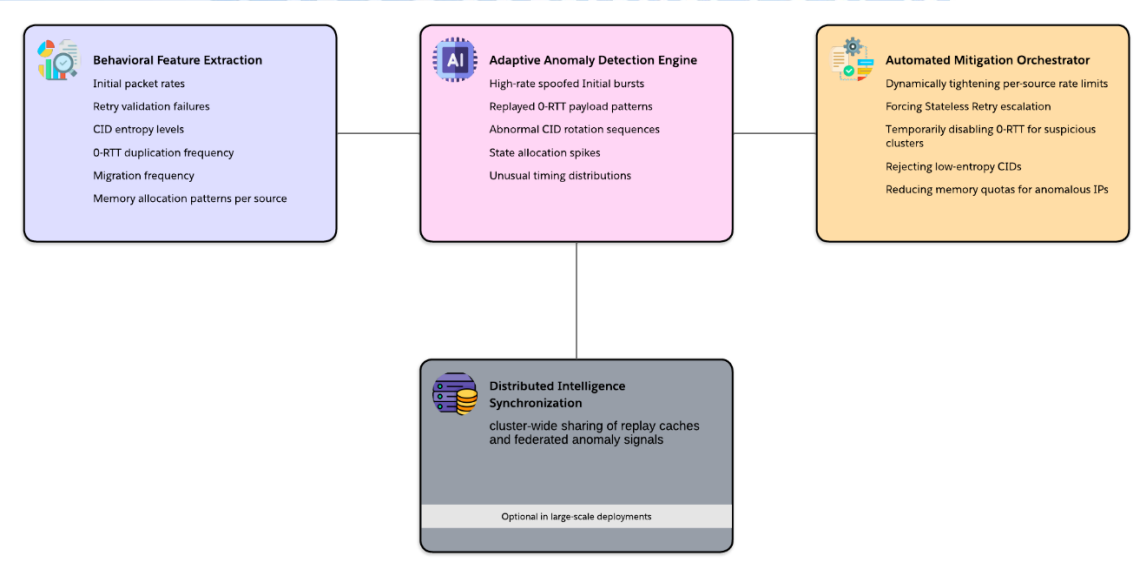

Figure 3: The Proposed AI-Augmented Defensive Architecture

The architecture of AI-assisted QUIC anomaly detection integrated into load balancers consists of 3 main layers, with an optional layer for large-scale deployments, as follows:

1- Behavioral Feature Extraction Layer

- Initial packet rates
- Retry validation failures
- CID entropy levels
- 0-RTT duplication frequency
- Migration frequency
- Memory allocation patterns per source

2- Adaptive Anomaly Detection Engine Layer

- High-rate spoofed Initial bursts
- Replayed 0-RTT payload patterns
- Abnormal CID rotation sequences
- State allocation spikes
- Unusual timing distributions

3- Automated Mitigation Orchestrator Layer

- Dynamically tightening per-source rate limits
- Forcing Stateless Retry escalation
- Temporarily turning off 0-RTT for suspicious clusters
- Rejecting low-entropy CIDs
- Reducing memory quotas for anomalous IPs

4- Optional Distributed Intelligence Synchronization Layer used in large-scale deployments operating in cluster-wide sharing of replay caches and federated anomaly signals to prevent attackers from distributing malicious traffic across multiple load balancer instances.

The model adopts a layered design consisting of the previously described layers.

4.3.4. Summary

While modern QUIC/HTTP3 protocols increase performance, they also bring about:

- Amplification exposure
- Replay risks
- Migration exploitation
- State exhaustion vectors
- Observability limitations

A non-trivial attack surface is created by implementation immaturity and protocol complexity, as evidenced by fuzzing studies, the exploitation of QUICforge, and research on cross-layer interactions. Instead of turning off QUIC, load balancers should be redesigned to:

- Enforce strict protocol validation
- Synchronize replay protection
- Route using CID awareness
- Integrate AI-driven encrypted traffic analysis.

This is in line with sophisticated cybersecurity models that integrate adaptive enforcement, behavioral modeling, and protocol compliance.

4.4. Including Zero Trust Concepts in the Architecture of Load Balancers

This section will answer the question “How can Zero Trust principles be integrated into load-balancing architecture?”

The main goals of traditional load-balancing architectures were scalability, availability, and performance. Perimeter firewalls or intrusion detection systems were usually tasked with enforcing security, implicitly relying on internal east-west traffic. Modern distributed cloud environments, however, refute this presumption. The National Institute of Standards and Technology (NIST) formally defined the Zero Trust Architecture (ZTA) model, which forbids implicit trust and requires ongoing authentication and authorization for each connection (Rose, OliverBorchert, Mitchell, & Connelly, 2020).

Additionally, new protocol-level behaviors, such as connection migration and encrypted transport metadata, are enabled by the adoption of QUIC and HTTP/3. Although these features enhance mobility and performance, they also pose a threat to conventional load-balancing techniques that rely on 5-tuple hashing)J. Iyengar و M. Thomson(2021 ،. To counter new attack vectors such as CID spoofing, migration storms, and state exhaustion, it is essential to incorporate Zero Trust principles directly into the load-balancing layer.

This section analyzes the security-latency trade-offs of an identity-aware Zero Trust load balancer architecture and proposes a Zero Trust architecture.

Recent research has emphasized the importance of Zero Trust architectural and policy-driven approaches to securing systems. (Rajendran, Rai, Anumula, & Agrawal, 2025) proposed a Zero Trust security model for micro services. The proposed model contains four components:

1- Identity Federation layer
   Responsible for authentication and authorization.
2- Workload Identity Management
   Provides short-lived service identities with mTLS between microservices.
3- Service Mesh Enforcement
   Responsible for network policies and traffic control.
4- Policy Engine
   A centralized agent for authorization across services, like the Kubernetes control plane.

The author claims that the design complies with the standards NIST SP 800-207 and SP 800-204. The proposed architecture in this section is considered robust because it adopts conceptually similar identity-centric, policy-driven, and least-trust principles that are consistent with these standards.

4.4.1. Load Balancing Using Zero Trust Design Principles

The following principles form the foundation of Zero Trust, according to NIST SP 800-207 (Rose, OliverBorchert, Mitchell, & Connelly, 2020):

1- All computing services or data sources are considered resources.
2- All communication is secured regardless of network location.
3- Access to individual resources is granted on a per-session basis.
4- Access to resources is determined by dynamic policy.
5- All assets are monitored, and their integrity and security posture are measured.
6- All resource authentication and authorization are dynamic and enforced before access.
7- All resources and network information are collected as much as possible to improve the security posture.

Because routing decisions are usually based on static IP-level hashing without identity verification, traditional L4 load balancers violate principles (2) and (4). This shows that the proposed architecture must be QUIC-based.

The suggested architecture overcomes this constraint by converting the load balancer into an identity-aware policy enforcement point that complies with Zero Trust specifications.

4.4.2. Proposed Architecture for a Zero Trust Load Balancer

The proposed architecture adheres to Zero Trust principles, as explained in the following sections.

Identity-Aware Edge Proxy with mTLS

The architecture implements mutual Transport Layer Security (mTLS) at the load balancing ingress. Workload identities (such as SPIFFE) or X.509 certificates are used for authentication in both client and backend services. This mechanism guarantees:

1- Authentication of cryptographic workloads.
2- Removal of implicit internal trust.
3- Preventing attacks caused by lateral movement.

Enforcing mTLS is consistent with service mesh security protocols used in platforms like Istio, where workload certificates are linked to identity. TLS 1.3 is incorporated directly into the transport handshake in QUIC-based systems)J. Iyengar و M. Thomson(2021 ،, enabling early identity validation without additional protocol layering.

Identity-Based Routing Mechanism

Routing must be restricted to validated identity rather than network metadata since QUIC uses Connection IDs (CIDs) to separate connection identity from IP/port. This stops CID manipulation and the exploitation of migration capabilities.

Traditional routing logic: hash (srcIP, dstIP, srcPort, dstPort, protocol)

Zero Trust routing logic proposal: hash (PolicyContext + VerifiedIdentity + ConnectionID)

The design enforces:

1- Validation of CID entropy.
2- Enforcing stateless retry before significant cryptographic allocation.
3- Identity-bound backend mapping.

According to QUIC security analyses, these controls reduce QUIC-specific amplification and migration abuse vulnerabilities)J. Iyengar و M. Thomson(2021 ، (Kakhki, Jero, Choffnes, Nita-Rotaru, & Mislove, 2019).

Policy Decision and Enforcement Layer

Similar to frameworks like Open Policy Agent, a load balancer incorporates a Policy Decision Point (PDP) to enable dynamic evaluation of:

1- Rules for service-to-service authorization.
2- Rate caps for each identity.
3- Thresholds for migration frequency.
4- Retry token validation compliance.

The Zero Trust principle is satisfied by evaluating policies on a session-by-session basis (3). As a result, the load balancer becomes an active authorization gateway instead of a passive traffic distributor.

Continuous Trust Evaluation via Behavioral Analytics

The architecture incorporates behavioral telemetry taken from QUIC sessions to expand Zero Trust beyond static authentication:

1- The initial rate of packets.
2- Retry validation errors.

3- Levels of CID entropy.
4- 0-RTT frequency of duplication.
5- Frequency of migration.
6- Patterns of memory allocation by source.

An AI-based anomaly-detection module processes these features, allowing for adaptive recalculations of trust. This is in line with Zero Trust's mandate for ongoing observation (Rose, OliverBorchert, Mitchell, & Connelly, 2020). As a result, trust is no longer binary but rather dynamic.

4.4.3. Proposed Architecture Design

The suggested Zero Trust-based load balancing architecture for QUIC/HTTP/3 environments is shown in Figure 4. According to the National Institute of Standards and Technology's Zero Trust model, the load balancer is conceptualized in the diagram as a centralized identity-aware security enforcement gateway rather than just a traffic distribution component (Rose, OliverBorchert, Mitchell, & Connelly, 2020).

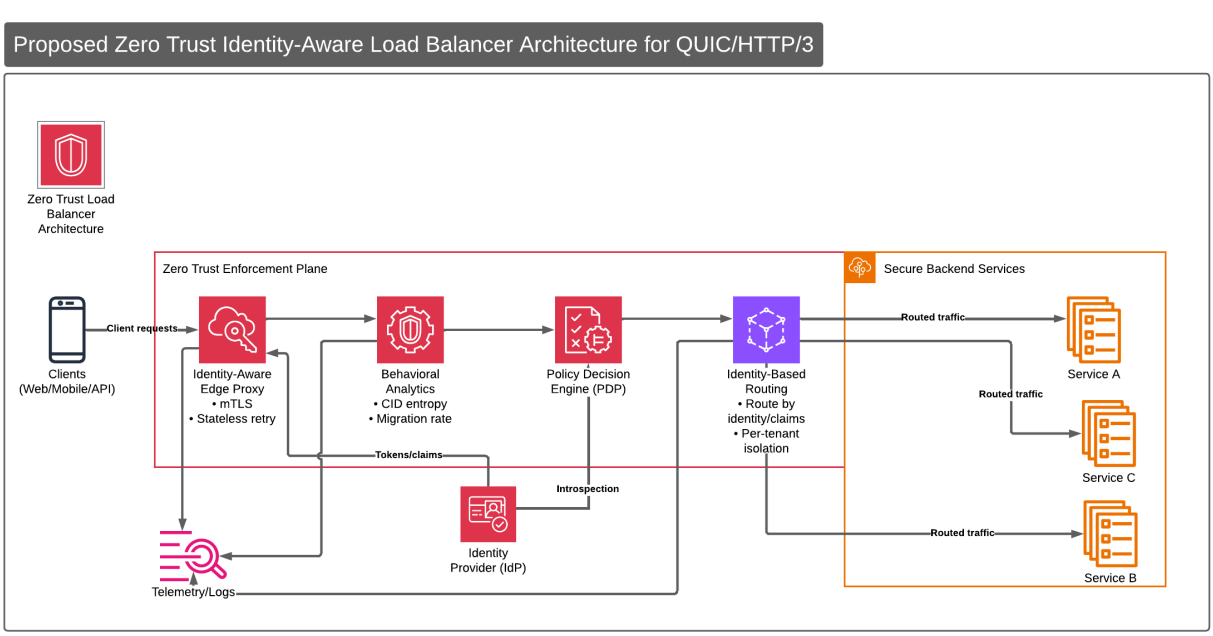

Figure 4: Proposed Zero Trust Identity-Aware Load Balancer Architecture for QUIC/HTTP/3.

The proposed design consists of the following:

1- Client Layer (Untrusted Zone)
All incoming traffic is treated as untrusted until cryptographically verified. No implicit trust is assumed at this stage.
2- Identity-Aware Edge Proxy (mTLS Enforcement Layer)
The Zero Trust principle, which states that all communication is secure regardless of network location, is upheld by this layer through:
- securely ends QUIC

- mutual TLS enforcement (TLS 1.3 integrated into QUIC)
- Verifies the workload identity or client certificates
- Before heavy state allocation, stateless retry tokens are issued.

3- Behavioral Feature Extraction Layer (AI Telemetry Layer)
This layer uses an AI-based anomaly scoring module that keeps an eye on:

- The initial rate of packets
- Retry validation errors
- Levels of CID entropy
- 0-RTT frequency of duplication
- Frequency of migration
- Patterns of memory allocation by source

4- Policy Decision and Identity-Based Routing Engine
Inspired by frameworks like Open Policy Agent, the Identity-Based Routing Engine communicates with a Policy Decision Point (PDP). Routing decisions are based on:

- confirmed the identity of the workload
- Policies for authorization
- Score for behavioral anomalies
- Validation of CID entropy
- Thresholds for migration

This upholds the Zero Trust tenets:

- Permission for each session
- Access with the least privileges
- Constant verification

5- Backend Service Layer (Trusted but Verified Zone)
Traffic is forwarded only to backend services whose identities and policies explicitly permit mTLS communication between the load balancer and the services, support Identity-bound service segmentation, and allow no implicit east–west trust.

4.4.4. Security–Latency Trade-Off Assessment

Zero Trust integration adds computational overhead even though it greatly improves security. Table 6 illustrates how the trade-offs can be categorized.

Table 6: Security–Latency Trade-Off in the Proposed Architecture.

Mechanism	Security Benefit	Latency Impact	Source of Overhead
mTLS Authentication	Prevents Lateral Movement	+1 RTT (Initial Handshake)	Certificate Verification
Policy Evaluation	Fine-Grained Authorization	Microseconds	Policy Lookup
AI Anomaly Scoring	Early Attack Detection	Microseconds – Milliseconds	Feature Processing
CID Entropy Validation	Prevents Spoofing or Migration Abuse	Minor Change	Bit-level Entropy Check

Compared to TCP+TLS combinations, QUIC's native 1-RTT handshake minimizes TLS overhead)J. Iyengar و M. Thomson(2021 ،. Additionally, token-based validation enables 0-RTT session resumption, reducing the cost of repeated handshakes.

Under typical operating conditions, cryptographic handshake overhead increases latency by 1-3%, according to empirical estimates from TLS performance studies (Rescorla, 2018). In return, the architecture greatly reduces the attack surface associated with state exhaustion and reflection amplification and eliminates implicit trust boundaries.

4.4.5. Summary

Routing decisions are made using an identity-based model rather than a network-based one when Zero Trust principles are incorporated into the load-balancing architecture. The suggested architecture routes traffic based on verified workload identity and ongoing behavioral evaluation, rather than solely on network information such as IP addresses or ports. This method helps reduce several QUIC-related risks, such as unauthorized east-west lateral movement, exploitation of connection migration, manipulation of Connection ID (CID) entropy, and handshake burst amplification.

While there is a slight increase in latency due to the cryptographic overhead of verification and policy evaluation, load balancing without implicit trust within the network helps tremendously reduce the attack surface from protocol-level weaknesses by enforcing identity-based access controls for workloads.

As illustrated in Figure 4.5.1, the proposed architecture has three contributions:

1- The load balancer serves as a Zero Trust Network Access (ZTNA) entry point.
2- It provides AI-driven behavioral monitoring at the transport layer.
3- It has identity-aware routing, blocking Connection ID spoofing, and connection migration attempts.

Concepts discussed in this section also apply to the Proposed AI-Augmented Defensive Architecture section under Protocol-Level Vulnerability Mitigations in QUIC-based HTTP/3, as well as to the Load Balancer Architecture section.

4.5. Defense-in-Depth Strategies for Kubernetes Ingress Controllers

This section will answer the question “What best practices and architectural patterns can mitigate security risks specific to Kubernetes ingress controllers and microservice environments?”

Kubernetes Ingress is a native API object that defines rules for routing external HTTP and HTTPS traffic to internal services within a cluster. Ingress uses host-based and path-based routing rules enforced by the ingress controller implementation, such as NGINX, Traefik, or an Istio Gateway, thereby creating a critical security boundary between untrusted external clients and internal micro services. Misconfigurations can lead to traffic manipulation (Kubernetes, 2025).

Kubernetes provides Transport Layer Security (TLS) for encrypting traffic and role-based access control (RBAC) as built-in security features for managing sensitive configuration data (Kubernetes, 2025). Kubernetes also provides Network Policies to control pod-to-pod communication, enabling microservice isolation (Kubernetes, 2025). Kubernetes implements ingress resources through a pluggable controller called “ingress controllers”, which is

responsible for routing rules and traffic logic (Kubernetes, 2025). These controllers vary significantly in their configuration, introducing both flexibility and risk.

The NGINX Ingress Controller recommends limiting RBAC permissions and restricting access to secrets and system files, aligning with Kubernetes best practices (NGINX, n.d.). Traefik reduces configuration complexity and automatically manages TLS, noting that default values require a clear revision (Labs, 2024). The Istio service mesh security model applies zero-trust principles by shifting trust from network location to service identity and using mutual TLS between services (Istio, 2024).

According to the Open Worldwide Application Security Project (OWASP) Kubernetes Top Ten list (OWASP, 2022), Kubernetes ingress security failures arise from misconfiguration rather than software vulnerabilities. The list includes multiple misconfigurations, including Insecure Workload Configurations, Overly Permissive RBAC Configurations, Missing Network Segmentation Controls, and Misconfigured Cluster Components. This encourages an experimental focus on configuration risks and how they will be mitigated.

Due to practical limitations in deploying all security scenarios in a virtualized Kubernetes environment, this section adopts a theoretical, design-driven analysis to examine each misconfiguration and produce an architectural mitigation pattern applicable across all microservice environments.

4.5.1. Microservice Misconfiguration Risks

Based on the previously reviewed official documentation and security guidance for Kubernetes and CNCF-aligned ingress technologies (NGINX Ingress Controller, Traefik, and Istio), Table 7 summarizes the identified risks and causes of misconfiguration.

Table 7: Common Microservice Risks with OWASP Mapping

Category	Risk	Misconfiguration Cause	OWASP Mapping
Ingress	Unintended Service Exposure	Wildcard Routing Rules	K9
Ingress	Traffic Interception	Missing TLS	K6
Microservice	Weak Service Authentication	Permissive mTLS Settings	K6
Ingress Controller	Privilege Escalation	Excessive RBAC Permissions	K3
Microservice	Lateral Movement	No Network Policies	K7

Unintended Service Exposure

Unintended service exposure occurs when ingress resources expose internal services, including administrative or backend APIs, to external users.

Traffic Interception

Traffic Interception is performed when ingress traffic is served over an unencrypted protocol, such as HTTP.

Weak Service Authentication

A Weak Service Authentication threat occurs during basic internal communication between services.

Privilege Escalation

Privilege Escalation refers to the use of excessive or default Role-Based Access Control (RBAC) permissions, allowing the ingress controller to escalate privileges or affect unrelated namespaces.

Lateral Movement

Lateral Movement is the unrestricted pod-to-pod communication. Network segmentation prevents lateral movement between pods.

4.5.2. Microservice Misconfiguration Mitigations

Due to practical limitations in deploying all security scenarios in a virtualized Kubernetes environment, this section adopts a theoretical, design-driven analysis to examine each misconfiguration. Table 8 summarizes the identified risks and their corresponding mitigation patterns.

Table 8: Common Microservice Risks with Mitigation Pattern

Category	Risk	Mitigation Pattern
Ingress	Unintended Service exposure	Explicit Routing
Ingress	Traffic Interception	Enforced HTTPS/TLS
Microservice	Weak Service Authentication	Strict mTLS, Zero Trust
Ingress Controller	Privilege Escalation	Least-Privilege RBAC
Microservice	Lateral Movement	Network Segmentation

Explicit Routing

According to previously reviewed documentation, best practices recommend that only intended services are externally accessible by explicitly defining the path and the host.

Enforced HTTPS/TLS

According to previously reviewed documentation, Ingress controllers rely on TLS termination to provide transport-level security using TLS certificates and redirect non-encrypted HTTP calls to encrypted HTTPS.

Strict mTLS with Zero Trust

Service meshes such as Istio use strict mutual TLS to ensure encryption and identity verification internally between services. mTLS applies zero-trust principles to pod-to-pod communication.

Least-Privilege RBAC

According to previously reviewed documentation, it is recommended to use least-privilege RBAC to restrict ingress controllers to only the resources required for routing and configuration, adding an extra security layer to the environment.

Network Segmentation

According to previously reviewed documentation, the Network Policies limit communication, ensuring that services can only communicate with explicitly authorized peers. The Network Policies also apply zero-trust principles to pod-to-pod communication.

4.5.3. Proposed Architecture Design

The identified risks highlight that ingress and microservice security are clearly affected by architectural and configuration decisions rather than implicit software flaws. This encourages the production of a security model that can be applied across different ingress technologies.

The illustration in Figure 5 shows the conceptual security architecture designed to mitigate common risks in ingress controllers and microservice environments. The proposed model addressed the previously mentioned risks, including unintended service exposure, traffic interception, privilege escalation at the ingress controller, lateral movement between microservices, and weak service authentication.

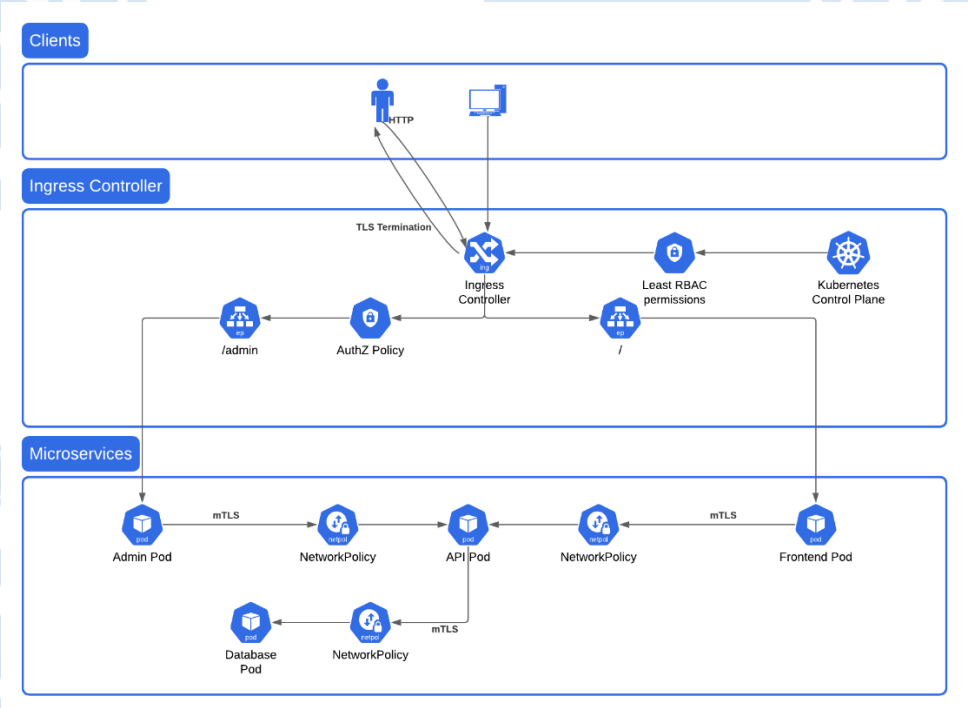

Figure 5: The Conceptual Security Architecture Design for Microservices

The Client Layer, Ingress Controller Layer, and Microservices Layer are the three main layers of the system.

The client Layer

The Client Layer represents external users or systems that access the application via HTTP-based protocols. As the cluster's only externally accessible entry point, the ingress controller is the sole means by which clients communicate with the platform.

The Ingress Controller Layer

The gateway between external clients and internal services is the Ingress Controller Layer. Before traffic reaches the microservices, the service is responsible for routing requests, terminating TLS, and enforcing access control. This layer mitigates the following:

1- Unintentional Service Exposure
To prevent unintentional exposure of internal or administrative services, authorization policies and routing rules ensure that only explicitly defined endpoints are accessible.
2- Traffic Interception
TLS termination prevents incoming traffic from being intercepted by encrypting communication between clients and the ingress controller.
3- Privilege Escalation
The ingress controller has the least-privileged Role-Based Access Control (RBAC) permissions when talking with the Kubernetes control plane. This is meant to prevent it from using administrative privileges and also ensure that attackers will not gain access to the rest of the cluster if the ingress component is compromised.

Microservices Layer

The frontend, API, admin, and database services are among the application components that are deployed as Kubernetes pods in the Microservices Layer. Network isolation and service authentication techniques are used to regulate communication between these services.

This layer mitigates the following:

1- Lateral Movement

Kubernetes Network Policies restrict communication routes between pods. If a service is compromised, attackers cannot move laterally across the cluster because only services expressly allowed to communicate with one another can do so.
2- Weak Service Authentication

Mutual TLS (mTLS) is used to secure service-to-service communication. This prevents malicious or unauthorized services from joining internal communications by ensuring that both communicating services authenticate each other.

4.5.4. Summary

In general, the design shown in Figure 1 illustrates how the ingress and service mesh layers minimize attack surface area and implement many zero-trust networking concepts found in microservice architectures. Design correctness, policy enforcement, and trust boundaries are some primary considerations with microservice security. The proposed architecture is similar to the Zero Trust security model for microservices, which was mentioned in the previously reviewed paper (Rajendran, Rai, Anumula, & Agrawal, 2025). As mentioned earlier, the author claims that the design complies with NIST SP 800-207 and SP 800-204. The proposed

architecture is considered robust because it adopts conceptually similar identity-centric, policy-driven, and least-trust principles that are consistent with these standards.

4.6. Key Findings

This chapter examines the experimental findings and the proposed architecture to address the study's research questions. The results show that the detection and mitigation of contemporary application-layer attacks can be enhanced by combining machine learning-based detection mechanisms with protocol-aware load balancing. The findings also emphasize the importance of protocol-level inspection and behavioral analysis in countering emerging threats such as HTTP/3- and QUIC-based attacks.

Overall, the results show that the suggested framework maintains a reasonable performance overhead while improving the robustness of the load-balancing environment. The assessment demonstrates that integrating intelligent detection mechanisms with protocol-level analysis is a viable approach to enhancing security in contemporary distributed systems.

The results of the experiment also emphasize the importance of combining traditional security measures with sophisticated analytical methods, empowering system architects to enhance their security strategies. The suggested method can find attack patterns that traditional rule-based systems might miss by utilizing behavioral monitoring and protocol-aware inspection. These findings imply that the overall defense posture of contemporary distributed applications can be considerably improved by incorporating adaptive security techniques into load balancer architectures.

5. Conclusion

This research demonstrated that load balancers can evolve from passive traffic distribution components into intelligent security enforcement layers within modern cloud infrastructures. By integrating machine learning, AI-driven Web Application Firewalls, protocol-aware defenses, and Zero Trust principles, the study showed that it is possible to enhance threat detection, mitigate protocol-level vulnerabilities, and enable identity-based routing decisions.

The findings confirm that machine learning models can effectively detect anomalous traffic patterns using load balancer telemetry, while AI-based WAFs improve detection accuracy compared to traditional approaches. Additionally, the analysis of HTTP/3 and QUIC highlighted critical vulnerabilities that can be mitigated through adaptive and behavior-based mechanisms at the load-balancing layer.

Despite limitations related to dataset scope, model complexity, and experimental environments, the proposed architecture provides a strong foundation for advancing secure load-balancing systems.

Future work should focus on leveraging advanced AI models, real-world datasets, distributed detection mechanisms, and full-scale implementation of Zero Trust and protocol-aware defenses. Overall, this study contributes to the growing body of research on cloud security by presenting a practical and forward-looking approach to embedding intelligent cybersecurity capabilities within load-balancing architectures.

6. References

Buchet, A., & Pelsser, C. (2025). A Study of Deployed Defenses Against Reflected Amplification Attacks in QUIC. *2025 9th Network Traffic Measurement and Analysis Conference*.

Chatzoglou, E., Kouliaridis, V., Kambourakis, G., Karopoulos, G., & Gritzalis, S. (2023). A hands-on gaze on HTTP/3 security through the lens of HTTP/2 and a public dataset. *Computers & Security*.

Chatzoglou, E., Kouliaridis, V., Karopoulos, G., & Kambourakis, G. (2022). Revisiting QUIC attacks: a comprehensive review on QUIC security and a hands-on study. *International Journal of Information Security*.

Chawla, K. (2024). Reinforcement Learning-Based Adaptive Load Balancing for Dynamic Cloud Environments. *ArXiv*.

Chentoufi, O., & Chougdali, K. (2021). Intrusion Detection Systems based on Machine Learning. *Proceedings of the 2nd International Conference on Big Data, Modelling and Machine Learning*.

Ed, M. T., & S. Turner, E. (2021). Using TLS to Secure QUIC. *Internet Engineering Task Force (IETF) RFC 9001*.

Gbur, Y., & Tschorsch, F. (2023). QUICforge: Client-side Request Forgery in QUIC. *NDSS*.

Ghomia, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*.

Grübl, T., Niu, W., Assen, J. v., & Stiller, B. (2025). QUIC-Exfil: Exploiting QUIC's Server Preferred Address Feature to Perform Data Exfiltration Attacks. *Proceedings of the 20th ACM Asia Conference on Computer and Communications Security*.

HAProxy. (n.d.). *HAProxy Documentation*. Retrieved 10 1, 2025, from https://docs.haproxy.org/

IEEE-USA. (2025). *AI: Providing New Challenges — and Solutions — in Cybersecurity*. Retrieved from insight: https://insight.ieeeusa.org/articles/ai-providing-new-challenges-and-solutions-in-cybersecurity/

Istio. (2024). *Security Best Practices*. Retrieved 1 2, 2026, from https://istio.io/latest/docs/ops/best-practices/security/

J. Iyengar, E., & M. Thomson, E. (2021). QUIC: A UDP-based multiplexed and secure transport. *Internet Engineering Task Force (IETF) RFC 9000*.

Kakhki, A. M., Jero, S., Choffnes, D., Nita-Rotaru, C., & Mislove, A. (2019). Taking a long look at QUIC: An approach for rigorous evaluation of rapidly evolving transport protocols. *Communications of the ACM*.

Katiyar, Nath, R., Pandey, Y., & Srivastava, P. (2025). *Intrusion Detection System Using Machine Learning Technique.* Retrieved from https://ssrn.com/abstract=5909202

Khare, S., Thakur, P., Talwandi, N. S., & Yadav, V. (2024). Securing Microservice Architecture: Load Balancing and Role-Based Access Control. *International Journal on Engineering Artificial Intelligence Management, Decision Support, and Policies*, 21-28.

Kubernetes. (2025). *Kubernetes Documentation*. Retrieved 1 1, 2026, from https://kubernetes.io/docs/home/

Kumar, G. (2014). Evaluation Metrics for Intrusion Detection Systems - A Study. *International Journal of Computer Science and Mobile Applications, 2*(11), 11-17.

Labs, T. (2024). *Traefik & Kubernetes*. Retrieved 1 2, 2026, from https://doc.traefik.io/traefik/reference/install-configuration/providers/kubernetes/kubernetes-ingress/

M. Bishop, E. (2022). HTTP/3. *Internet Engineering Task Force (IETF) RFC 9114*.

Mahmoudi, N., & Khazaei, H. (2022). MLProxy: SLA-Aware Reverse Proxy for Machine Learning Inference Serving on Serverless Computing Platforms. *ArXiv*.

Microsoft Defender Security Research Team. (2020). *Seeing the big picture: Deep learning-based fusion of behavior signals for threat detection*. Retrieved from Microsoft Security Blog: https://www.microsoft.com/en-us/security/blog/2020/07/23/seeing-the-big-picture-deep-learning-based-fusion-of-behavior-signals-for-threat-detection/

MK, A., Bala, K. S., Sonti, S. S., & KP, J. (2025). An Empirical Study on the Evaluation and Enhancement of OWASP CRS (Core Rule Set) in ModSecurity. *Computers & Security*.

ModSecurity. (n.d.). *ModSecurity Documentation*. Retrieved 12 1, 2025, from https://modsecurity.org/

Moharir, C., Kuppuraju, S. Y., & Kumar, N. (2025). Advanced Load Balancing Strategies using AI to Optimize Cloud Service Delivery. *International Journal of All Research Education and Scientific Methods (IJARESM), 13*(3).

Mouzakitis, E. (2015). *Monitoring HAProxy performance metrics.* Retrieved from https://www.datadoghq.com/blog/monitoring-haproxy-performance-metrics

Nawaz, M., Tahira, S., Shah, D., Ali, S., & Tahir, M. (2025). Lightweight machine learning framework for efficient DDoS attack detection in IoT networks. *Sci Rep, 4*. doi:https://doi.org/10.1038/s41598-025-10092-0

Networks, F. (2023). *What is an Application Delivery Controller (ADC)?* Retrieved from https://www.f5.com/glossary/application-delivery-controller

Networks, F. (n.d.). *Load Balancing 101: Nuts and Bolts*. Retrieved from https://www.f5.com/resources/white-papers/load-balancing-101-nuts-and-bolts

Networks, F. (n.d.). *What Is a Load Balancer?* Retrieved from https://www.f5.com/glossary/load-balancer

NGINX. (n.d.). *NGINX Documentation*. Retrieved 12 1, 2025, from https://nginx.org/en/docs/

Otero-Mosquera, J., Lo´pez-Bravo, C., Tub´ıo-Figueira, P., & Iglesia, A. I. (2025). Improving WAF Detection Capabilities Through Machine Learning Algorithms in Open-Source Technologies. *Security and Communication Networks*.

OWASP. (2022). *OWASP Kubernetes Top 10*. Retrieved 1 2, 2026, from https://owasp.org/www-project-kubernetes-top-ten/

Prometheus. (n.d.). *Prometheus Documentation*. Retrieved 10 1, 2025, from https://prometheus.io/docs/introduction/overview/

Rajendran, R. N., Rai, D. K., Anumula, S. K., & Agrawal, S. (2025). Zero Trust Security Model Implementation in Microservices Architectures Using Identity Federation. *arXiv*.

Rescorla, E. (2018). The transport layer security (TLS) protocol version 1.3. *Internet Engineering Task Force (IETF) RFC 8446*.

Rose, S., OliverBorchert, Mitchell, S., & Connelly, S. (2020). Zero Trust Architecture. *NIST special publication*. Retrieved from https://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-207.pdf

Sengupta, J., Kosek, M., Fries, J., Ferlin-Reiter, S., & Bajpai, V. (2024). On cross-layer interactions of QUIC, encrypted DNS and HTTP/3: design, evaluation, and dataset. *IEEE Transactions on Network and Service Management*.

Sivaraman, H. (2024). ML-Based Threat Detection for Container Network Security in Kubernetes. *International Journal of Innovative Research and Creative Technology, 10*(1), 1-8. doi:https://doi.org/10.5281/zenodo.14250606

Stardom University

Stardom Scientific Journal of Natural and Engineering Sciences

- Stardom Scientific Journal of Natural and Engineering Sciences -
Peer Reviewed Scientific Journal published twice
a year by Stardom University

2nd issue- 4th Volume 2026

ISSN 3756-2980