Anomaly Detection in Encrypted Web Traffic Using Deep Learning Mechanisms

By

Andimir'irenge Gubanja Albert

153079

Submitted in Partial Fulfilment of the Requirements for the Degree of Master of Science in Information Systems Security at Strathmore University

School of Computing & Engineering Sciences

Strathmore University

Nairobi, Kenya

June, 2025

Declaration and Approval

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the dissertation contains no material previously published or written by another person except where due reference is made in the dissertation itself.

Student's Name: Andemir'irenge Gubanja, Albert

Signature …… Date…May 21, 2025……

Approval

The dissertation of Andemir'irenge Gubanja Albert was reviewed and approved by the following:

Dr. John Olukuru
Head of Data Science and Analytics
Director, Risk Management Centre

Dr. Julius Butime,
Dean, School of Computing & Engineering Sciences,
Strathmore University

Prof. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University

Table of Content

Table of Figures

List of Tables

List of Abbreviations

AEs	Autoencoders
ALPN	Application-Layer Protocol Negotiation
AUC	Area Under the Curve
C2	Command-and-control
CIC	Canadian Institute for Cybersecurity
CICIDS	Canadian Institute for Cybersecurity Intrusion Detection System
CNNs	Convolutional Neural Networks
CSV	Comma-Separated Values
DCGAN_1D-CNN	Deep Convolutional Generative Adversarial Networks with 1D Convolutional Neural Networks
DoS	Denial of Service
DPI	Deep Packet Inspection
DSR	Design Science Research
DTLS	Datagram Transport Layer Security
F1-score	Harmonic mean of precision and recall
FP	False Positive
FPGA	Field-Programmable Gate Array
GDPR	General Data Protection Regulation
GRUs	Gated Recurrent Units
HIPAA	Health Insurance Portability and Accountability Act
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IDPS	Intrusion Detection and Prevention Systems
IDS	Intrusion Detection System
IP	Internet Protocol
IPFIX	IP Flow Information Export
IPsec	Internet Protocol Security
IQR	Interquartile Range
JA3	**J**ohn Althouse, Jeff Atkinson, and Josh Atkins
JSON	JavaScript Object Notation
KNN	K-Nearest Neighbours
LFI	Local File Inclusion

LSTM	Long Short-Term Memory
MAC	Media Access Control
MD5	Message Digest Algorithm 5
MiTM	Man-in-the-Middle
MLP	Multi-Layer Perceptron
MTBD	Multi-step Traffic Behaviour Detection
NIDS	Network Intrusion Detection Systems
NTP	Network Time Protocol
PCAP	Packet Capture
RFI	Remote File Inclusion
S/MIME	Secure/Multipurpose Internet Mail Extensions
SIEM	Security Information and Event Management
SNI	Server Name Indication
SPAN	Switch Port Analyser
SSH	Secure Shell
SSL	Secure Sockets Layer
TLS	Transport Layer Security
UNSW-NB15	University of New South Wales - Network Benchmark Dataset 15
VLAN	Virtual Local Area Network

Dedication

This research is dedicated to the relentless pursuit of innovation in cybersecurity, and to the countless hours spent defending our digital infrastructure.

Acknowledgements

This research was successfully completed due to the contributions of key individuals and institutions. I extend my sincere gratitude to Dr. John Olukuru, my supervisor, for his expert guidance and continuous support throughout this project.

I also acknowledge the valuable contributions of Dr. Joseph Sevilla and Dr. Vitalis Gavole, as well as the entire staff of Strathmore University, particularly those within @iLabAfrica and the Faculty of Information Technology. Their provision of a conducive research environment and technical resources significantly aided in the execution of this project.

Finally, I express my appreciation to my colleagues and peers for their collaborative spirit and intellectual exchange.

Abstract

The increasing adoption of HTTPS encryption has significantly improved data confidentiality but has simultaneously challenged traditional anomaly detection systems that rely on content-based or signature-based inspection. This research addresses the critical problem of detecting anomalies in encrypted web traffic without decrypting the data, by leveraging deep learning techniques that operate solely on metadata and behavioural features.

The study proposed and developed a dual-model anomaly detection system using supervised deep learning. Two classification models were trained independently: one on packet-based features and another on session-based features extracted from structured HTTPS datasets containing both benign and malicious traffic. The models achieved test accuracies of 98.64% and 99.66%, respectively. When combined using a voting-based decision strategy, the overall system achieved an exceptional detection accuracy of 99.995%.

The methodology followed a Design Science Research (DSR) approach, encompassing problem identification, model development, and experimental validation. Although real-time deployment was not achieved due to hardware and infrastructure constraints, the system was successfully implemented and tested in a simulated environment, validating its feasibility and effectiveness.

This research demonstrates that it is possible to detect encrypted traffic anomalies accurately and efficiently using metadata alone, without compromising privacy. The results lay the groundwork for future development of scalable, real-time, and privacy-preserving intrusion detection systems suitable for modern encrypted network environments.

Keywords: HTTPS, Deep learning, Anomaly detection, Encrypted web traffic, DSR

Definition of Terms

i. **Anomaly Detection**: Anomaly detection is the process of identifying data points or behaviours that deviate from the expected norm. In cybersecurity, it is crucial for detecting unusual network activity that may indicate potential threats, such as cyberattacks or unauthorized access attempts (Chandola et al., 2009).

ii. **Encrypted Web Traffic**: Encrypted web traffic refers to internet data that has been encoded using cryptographic methods to ensure privacy and security. Instead of being transmitted in plain text, the data is transformed into an unreadable format that only authorized recipients can decrypt (Rescorla & Dierks, 2008). This encryption protects sensitive communications from interception and tampering by attackers.

iii. **Deep Learning**: Deep learning is a subset of artificial intelligence that enables computers to process data in a way that mimics human learning. By using multiple layers of neural networks, deep learning models can recognize complex patterns, make predictions, and improve decision-making based on large datasets (LeCun et al., 2015).

iv. **Cybersecurity Threats**: Cybersecurity threats encompass risks that could compromise the confidentiality, integrity, or availability of digital systems. These threats include malware, phishing, ransomware, and other attack vectors that exploit vulnerabilities in networks, applications, or human behaviour (Pfleeger & Pfleeger, 2012).

v. **HyperText Transfer Protocol Secure (HTTPS)**: HTTPS is the secure version of HTTP, the communication protocol that enables web interactions. It relies on encryption protocols such as SSL/TLS to protect data exchanged between web browsers and servers (Rescorla, 2018).

vi. **Deep Packet Inspection (DPI)**: Deep Packet Inspection is an advanced network traffic analysis technique that examines not only packet headers but also the actual content of network packets. This enables security systems to detect malware, enforce access policies, and identify application usage (Roesch, 1999).

vii. **Autoencoders (AEs)**: Autoencoders are unsupervised neural networks designed to learn efficient representations of input data by reconstructing it with minimal error. They are widely used in anomaly detection because they can capture the underlying structure of normal network traffic and flag deviations that may indicate potential threats (Hinton & Salakhutdinov, 2006).

viii. **Packet Flow Characteristics**: Packet flow characteristics refer to measurable properties of data transmission, such as packet size, frequency, direction, and timing. These attributes play a critical role in network traffic analysis, enabling the distinction between normal and suspicious activities (Moore & Zuev, 2005).

ix. **Command-and-Control (C2) Communications**: Command-and-Control (C2) communications refer to covert channels used by cybercriminals to manage compromised systems remotely. Attackers utilize C2 infrastructure to issue commands, extract stolen data, and coordinate malicious activities such as botnet operations (*Command and Control (C&C) Attacks*, n.d.). Detecting and disrupting C2 traffic is a key strategy in mitigating cyber threats, as it prevents attackers from maintaining control over infected devices.

x. **Design Science Research (DSR) Methodology**: Design Science Research (DSR) is a problem-solving research methodology that focuses on developing and evaluating innovative solutions to practical challenges. In cybersecurity, DSR is employed to design, build, and assess new frameworks, such as deep learning-based anomaly detection systems (Hevner et al., 2004).

xi. **Hyperparameter Optimization**: Hyperparameter optimization is the process of fine-tuning machine learning model parameters to improve performance. Unlike model parameters, which are learned from data, hyperparameters—such as learning rate, number of layers, and activation functions—must be set manually or through automated search techniques like grid search and Bayesian optimization (Bergstra & Bengio, 2012).

Chapter 1 : Introduction

1.1 Background

The accelerated evolution of technology and the extensive adoption of the Internet are transforming how we communicate, conduct business, and manage information (Agarwal et al., 2022). Cybersecurity remains a crucial issue as each technological advancement is accompanied by new threats. Protecting information systems against cybercriminals and malicious activities is an ongoing concern. Indeed, as soon as new security measures are implemented, new techniques to circumvent them emerge (Pandey et al., 2022; Zaid & Garai, 2024). Thus, it is essential to design and implement innovative methods to enhance system security, particularly the security of data transmitted over networks.

Detecting malicious activities or anomalies on networks is fundamental to stopping, capturing, or destroying them. Techniques such as firewalls and Intrusion Detection and Prevention Systems (IDPS) have been developed for this purpose: to detect, prevent, stop, and correct threats (Thapa & Mailewa, 2020). However, these measures are not always sufficient to guarantee information security (Sadqi & Mekkaoui, 2021). Data encryption has therefore become an advanced solution to ensure the confidentiality and integrity of data in transit on the Internet (Ma & Zhang, 2020).

Numerous encryption protocols are used to secure information at different levels of network communications. Among these protocols, HyperText Transfer Protocol (HTTP) over Secure Sockets Layer / Transport Layer Security (SSL/TLS), which operates at the application layer, ensures that data exchanged between a web browser and a web server remain confidential and intact. Other notable encryption protocols include SSL/TLS, Internet Protocol Security (IPsec), Secure Shell (SSH), Secure/Multipurpose Internet Mail Extensions (S/MIME), and Datagram Transport Layer Security (DTLS) (Seth et al., 2022).

However, despite the significant advantages of these encryption techniques, they can also be exploited by cybercriminals to mask their malicious activities. Encryption makes threat detection more difficult, as security tools cannot easily inspect encrypted traffic without violating privacy (B. Wang et al., 2022). Recent research has shown that many attackers now use encrypted communications to distribute malware, making it harder to detect using traditional methods (Singh & Singh, 2021).

This challenge has led to the exploration of artificial intelligence, which has demonstrated remarkable capabilities in cybersecurity. By analysing metadata and traffic patterns rather than decrypting data, AI-driven deep learning models can identify anomalies that indicate potential cyber threats. This research specifically focuses on developing such mechanisms for HTTPS traffic, ensuring enhanced detection capabilities while preserving data confidentiality.

1.2 Problem Statement

The core problem of this research is the increasing vulnerability of organizational cybersecurity due to the exploitation of encrypted HTTPS traffic by cybercriminals. While encryption is essential for securing data in transit, it also provides a secure conduit for malicious activities, including malware distribution, covert data exfiltration, sophisticated phishing attacks, and stealthy command-and-control communications. Traditional anomaly detection systems, which rely on payload inspection, are rendered ineffective against these encrypted threats. Indeed, cybercriminals exploit encrypted channels to bypass security measures and carry out attacks undetected, leveraging the trust encryption protocols inherently provide. Network security administrators are therefore faced with the critical challenge of detecting and mitigating these threats without compromising data privacy or incurring significant performance overhead through decryption. This gap highlights the need for alternative detection approaches that can analyse encrypted traffic without decrypting it.

1.3 Research Objective

The main objective of this research is to develop a deep learning-based anomaly detection system for encrypted web traffic that ensures the resilience of web applications against evolving threats by accurately identifying and detecting threats while preserving the privacy and security benefits of encryption.

1.3.1 Specific Objectives

1. Gain a comprehensive understanding of the nature of cybersecurity threats prevalent in encrypted web traffic data and identify their characteristics and patterns.
2. Review and assess existing literature and solutions to pinpoint gaps and limitations in current approaches for encrypted web traffic anomaly detection.
3. Design and develop a novel deep learning-based model optimized for encrypted web traffic anomaly detection.

4. Validate the practical applicability and reliability of the deep learning-based anomaly detection system in enhancing web security and reducing vulnerabilities.

1.4 Research Questions

1. What are the characteristics and behavioural patterns of cybersecurity threats prevalent in encrypted web traffic data?
2. What are the gaps and limitations in existing solutions for anomaly detection in encrypted web traffic?
3. Which deep learning models and algorithms are suitable for developing an effective anomaly detection system for encrypted web traffic?
4. How can the developed deep learning-based anomaly detection system demonstrate practical utility in enhancing web security?

1.5 Research Hypothesis

Given the objectives and research questions outlined, the following hypothesis is proposed:

A deep learning-based anomaly detection system, utilizing metadata analysis of encrypted web traffic, can achieve a significantly higher accuracy rate in detecting cybersecurity threats compared to traditional signature-based methods, while preserving data privacy.

Deep learning models are capable of extracting complex patterns from network traffic metadata (e.g., packet size, timing, flow characteristics) without requiring decryption. This allows for the identification of anomalous behaviours indicative of malicious activities, such as malware distribution and data exfiltration, which are often concealed within encrypted HTTPS traffic.

1.6 Scope and Limitations

This research focuses on developing and evaluating a deep learning-based anomaly detection system for encrypted HTTPS traffic, specifically targeting malware distribution, data exfiltration, phishing, and command-and-control communications. The system analysed traffic patterns and metadata using deep learning models (autoencoders) without decrypting content, thereby preserving privacy. Validation was conducted using real-world encrypted traffic datasets, assessing detection accuracy, false positive rates, and processing efficiency. The research also considered the integration with existing security infrastructures.

This study is limited to anomaly detection within the application layer of HTTPS traffic and excludes other network protocols like IPsec, SSL/TLS, or SSH, as well as non-web application encrypted traffic (e.g., email). The primary focus is on a proof-of-concept; therefore, deployment and scalability in large-scale production environments will require further refinement. Generalizability may be impacted by regional differences in internet usage and network infrastructure, and access to diverse real-world encrypted traffic datasets can be challenging. Controlled testing environments may not fully replicate real-world network conditions. Finally, the research focuses solely on anomaly detection, not on response or mitigation strategies.

1.7 Justification of the Research

This research is critical due to the increasing sophistication of cybersecurity threats within encrypted HTTPS traffic, which traditional security measures struggle to detect. By developing a deep learning-based anomaly detection system, this study aims to enhance real-time threat detection capabilities, strengthening overall cybersecurity defences.

A key focus is preserving the privacy inherent in encryption while enabling effective threat detection. The research leverages encrypted traffic patterns and metadata to identify anomalies without decryption, balancing security and user privacy. This is vital for organizations across sectors like finance, healthcare, and government, where secure web communication is paramount.

This work advances deep learning techniques in cybersecurity by exploring and optimizing autoencoders models for encrypted traffic analysis. It bridges a gap in existing literature, providing new methodologies for addressing anomaly detection in encrypted web traffic. Practically, it offers a tangible solution for cybersecurity practitioners to enhance threat defence.

Ultimately, this research aims to create a safer digital environment by improving detection capabilities while respecting privacy. The following literature review will demonstrate the current state of research, identifying gaps and motivating this study.

Chapter 2 : Literature Review

2.0 Introduction

The rapid advancement of technology and the widespread adoption of encryption protocols have transformed the landscape of web traffic, presenting both new opportunities and challenges in the realm of cybersecurity. This literature review aims to synthesize existing research on anomaly detection within encrypted HTTPS traffic, focusing on how previous studies have approached the problem of identifying cybersecurity threats concealed within encrypted channels. By examining various methodologies and their results, this review highlighted the effectiveness of different anomaly detection techniques and the insights gained from previous findings.

2.1 Cybersecurity Threats in Web Traffic

Grammatikis and Sarigiannidis (2021) classify cybersecurity threats into four categories, namely DoS attacks, Routing Attacks, Network Traffick Analysis and MiTM attacks, and Web application attacks. They distinguish between threats targeting web applications and others, which target bandwidth, routing protocols, and network traffic, respectively. In this context, they consider a web security threat or issue as a potentially malicious action specifically targeting certain elements of a web application's architecture, such as the user's browser or the server hosting the application. Through their analysis, they identified five main threats targeting web application: malicious proxy attacks, SQL injection, local file inclusion (LFI), remote file inclusion (RFI), and command execution attacks. These threats are directly relevant to this research, as they represent the most common known anomalies that many web anomaly detection systems aim to address. In this study, these known threats, along with other potential unknown anomalies, will be analysed using the capabilities of deep learning for enhanced detection and mitigation.

2.2 Anomaly Detection Definition

Anomaly detection in computer systems is not a new technique and has already been the subject of several studies. Patcha and Park (2007) provided a more specific cybersecurity-focused definition of anomaly detection. They defined it as the identification of activities or behaviours within a network that deviate from the expected norms, which can signal potential intrusion attempts or security breaches. They also emphasized the importance of anomaly detection in maintaining the integrity and security of computer networks. Their research provides a key

linkage to this study, particularly in understanding how anomaly detection can be adapted to encrypted web traffic environments where traditional inspection techniques are ineffective. The need to detect deviations without accessing encrypted content is a central challenge of this research.

Chandola et al. (2009) provided a foundational definition of anomaly detection, describing it as the identification of patterns in data that do not conform to expected behaviour. They trace its origins to the need to protect systems from malicious activities by identifying deviations from normal functioning. Their review covered various domains, including cybersecurity, and emphasized that anomalies often represent potential security threats such as intrusions or system failures. This research builds upon their definition by specifically focusing on anomaly detection within encrypted web traffic using deep learning to identify deviations in behaviour that may signal cyberattacks.

Similarly, Bindu and Thilagam (2016) explored anomaly detection in the context of intrusion detection systems (IDS). Their work focused on identifying unusual network behaviour, which they described as suspicious events deviating from normal traffic patterns. They used traditional machine learning techniques to detect these anomalies, highlighting that early detection of deviations can significantly improve network security. While their work mainly utilized rule-based and signature-based methods, this research expands on their findings by applying deep learning techniques, which allow for the detection of anomalies in encrypted traffic—an area where traditional methods struggle.

2.3 Methods for Anomaly Detection

Anomaly detection in computer systems and networks is a practice that has evolved over time. Patcha and Park (2007) examined the limitations of advanced statistical models used in anomaly detection. They noted that while time-series analysis and deviation detection methods were introduced, these models were still prone to false positives and false negatives, especially when applied to large and dynamic datasets. This is particularly significant for encrypted traffic, where identifying abnormal trends without decryption requires more sophisticated methods, which this research aims to address using deep learning.

Shaukat Dar et al. (2021) explored early methods of anomaly detection, which primarily relied on manual approaches, including visual inspection of system logs and basic statistical techniques. Their study demonstrated that while human observation and simple statistics, such

as means and standard deviations, could sometimes identify anomalies, these methods were prone to human error and could not handle large volumes of data in real-time. They found that these basic methods were often inefficient for complex datasets, leading to numerous false positives. This is relevant to the current research, as it highlights the limitations of manual anomaly detection, especially when dealing with encrypted traffic, which significantly increases data complexity.

Similarly, Pang et al. (2022) reviewed statistical methods for anomaly detection in network systems, focusing on how basic metrics such as averages and distributions were used to detect irregularities in earlier systems. Their findings suggest that while statistical methods could identify extreme deviations from normal behaviour, they often failed to detect subtle anomalies, particularly in large-scale or encrypted data environments. This limitation is particularly pertinent to encrypted web traffic, where anomalies may be obscured by encryption layers, making traditional statistical approaches inadequate for detecting sophisticated threats.

Schmidl et al. (2022) also focused on the evolution of manual anomaly detection methods. Their study highlighted the introduction of Intrusion Detection Systems (IDS) and advanced statistical models in the 1970s and 1990s, where anomaly detection shifted towards more structured, automated approaches. IDS were designed to detect known attack signatures. However, they found that IDS were unable to detect unknown or evolving threats, as these systems required frequent signature updates. This is a critical insight for this research, which focuses on detecting unknown anomalies in encrypted web traffic, where new threats are constantly emerging, and signature-based detection is often insufficient.

Rosa et al. (2021) explored network traffic analysis and rule-based anomaly detection systems, which became prominent in the 1990s. They found that while network monitoring tools and auditing systems were useful in identifying abnormal behaviours like unauthorized connections or traffic spikes, they were overwhelmed by the volume of data in modern networks, especially with encrypted traffic. Rigid rule-based systems, according to Rosa et al., lacked the flexibility to adapt to new or evolving threats. This limitation aligns with the challenges of encrypted traffic anomaly detection, where predefined rules may not account for the complexities introduced by encryption, further underscoring the need for more adaptive, machine learning-based solutions.

Lastly, Tulbure et al. (2022) explored more recent advancements in anomaly detection, particularly the application of clustering algorithms (like k-means) and early artificial neural networks. Their study found that clustering methods could group similar behaviours and detect outliers, but their performance was highly dependent on parameter configurations and the quality of training data. Early neural networks, while promising, required extensive computational resources and large training datasets, making them difficult to deploy in real-time systems. These insights are particularly relevant to this research, as deep learning techniques, while powerful, also face challenges in terms of computational requirements and model interpretability, especially when applied to the complex problem of detecting anomalies in encrypted HTTPS traffic.

2.4 Anomaly Detection in Encrypted Traffic

The encryption of data has significantly altered the landscape of anomaly detection, as traditional methods, previously dependent on inspecting unencrypted traffic, have become ineffective in identifying threats. Encryption, while enhancing data security by ensuring confidentiality and integrity, also obscures network traffic content, presenting challenges for detection systems. As systems increasingly adopt encryption, anomaly detection methods have had to evolve, with researchers focusing on machine learning and deep learning approaches to address the limitations of traditional techniques.

Vikram and Mohana (2020) sought to address the challenges of detecting anomalies in encrypted traffic by enhancing traditional Intrusion Detection Systems (IDS) with machine learning techniques. They applied the Isolation Forest algorithm, trained on the KDD dataset, to detect anomalies in network traffic. Their findings indicated that machine learning could significantly improve the detection of anomalies compared to signature-based IDS. However, while effective, the study did not specifically address encrypted traffic, which is critical to this research. The relevance of their work lies in its demonstration of the potential of machine learning in evolving IDS capabilities, providing a foundation for more complex models applicable to encrypted traffic.

Similarly, Goel et al. (2022) compared various supervised machine learning algorithms, including Random Forest and Decision Trees, to assess the accuracy of IDS. They found that Random Forest achieved the highest accuracy of 99.84%. While the study showcases the success of machine learning for anomaly detection in unencrypted traffic, the use of supervised

learning models presents limitations when applied to encrypted traffic. Encrypted traffic poses challenges for supervised models, as labelled data is often scarce. This limitation aligns with the focus of this research, which aims to move beyond supervised learning towards more adaptive deep learning techniques suitable for encrypted traffic.

Sah and Venkatesh (2024) further explored machine learning methods for intrusion detection in network traffic, utilizing Decision Trees and Random Forests. Their investigation, based on the Machine Learning CSV dataset from the CICIDS-2017 dataset, achieved a detection accuracy of 99%. While their work demonstrates high accuracy, it is limited to unencrypted data. The methods, though effective, struggle with the complexities of encrypted traffic, making their contribution relevant to this research as a basis for advancing towards deep learning models capable of handling encrypted data.

In their study, Isingizwe et al. (2021) applied AutoML to classify encrypted malware traffic, utilizing a pipeline of seven models with automated hyper-parameter tuning. Their findings revealed strong performance in detecting encrypted malware traffic, with the identification of key discriminant features, especially within TLS metadata. Despite the model's ability to reduce false positives and false negatives, it could not detect new attack patterns, which is critical in environments where cyber threats continuously evolve. This highlights a key challenge that this research aims to address by incorporating deep learning methods capable of detecting novel threats in encrypted traffic.

On the other hand, Long and Zhang (2023) critically analysed the limitations of traditional machine learning models in detecting anomalies, particularly in feature selection. Their work demonstrated that machine learning models struggle with adapting to new features in encrypted traffic due to their reliance on predefined training data. This study is closely related to this research, as it emphasizes the need for more adaptive models that can handle dynamic and encrypted traffic environments. They advocate for deep learning approaches that can better learn from encrypted data without requiring extensive feature engineering.

Yu et al.(2019) presented a deep learning-based anomaly detection system using autoencoders, which showed promising results on encrypted traffic. Their study utilized multiple autoencoders to detect deviations from normal behaviour in network traffic, achieving high accuracy. This research directly builds on such findings by extending the use of autoencoders in anomaly detection within encrypted HTTPS traffic. The potential of autoencoders to handle

encrypted data efficiently makes this approach suitable for this research's focus on detecting unknown anomalies in complex traffic environments.

Xing and Wu (2020) introduced D2LAD, an anomaly detection system based on deep dictionary learning, designed to address the shortcomings of traditional packet inspection methods in encrypted traffic. Their model achieved an accuracy of 94.5%, showing significant improvements in handling large volumes of data. By focusing on sequential feature extraction and deep learning, their approach provides a strong foundation for this research, which seeks to apply similar deep learning techniques to the domain of HTTPS traffic, ensuring scalability and high accuracy in detecting encrypted anomalies.

Yang and Lim (2021) proposed an LSTM-based autoencoder for detecting anomalies in SSL traffic, addressing limitations in traditional IDS and DPI methods. Their approach involved reducing noise and enhancing feature extraction in high-dimensional data, producing image-type inputs for a CNN classifier, which then classified traffic as malicious or benign. This method aligns with the goals of this research, which aims to use deep learning techniques like autoencoders to improve the detection of anomalies in encrypted web traffic, particularly HTTPS.

Behdadnia et al. (2023) focused on early anomaly detection in encrypted traffic over electric networks. Their deep learning-based model, combining CNN and byte-to-image transformation, achieved a 93% accuracy rate in detecting anomalies in IPsec-secured networks. This work is relevant to the current research as it explores deep learning's potential in encrypted environments, although its focus on IPsec protocols differs from this research's focus on HTTPS traffic.

Cheng et al. (2023) made significant contributions by using an improved CNN model to detect malicious traffic in SSL/TLS traffic, enhancing detection accuracy by 0.0761 and demonstrating robustness in handling encrypted data. Their work supports the exploration of CNNs in detecting anomalies within encrypted web traffic, highlighting the importance of robust feature extraction techniques, which are integral to this research's methodology.

Bahlali et al. (2023) emphasized the shortcomings of traditional NIDS and machine learning techniques for anomaly detection in encrypted traffic. Their study advocates for the adoption of deep learning methods, demonstrating that autoencoders, CNNs, and other neural network

models significantly outperform traditional methods. This aligns directly with this research, which aims to develop a deep learning-based anomaly detection system tailored for encrypted HTTPS traffic, leveraging autoencoders to improve accuracy and reduce false alarms.

Long and Zhang (2023) introduced the Deep Encrypted Traffic Detection (DETD) framework, which uses a multi-layer autoencoder for feature extraction and L1 regularization for feature selection. DETD achieved an anomaly detection performance of 99.998%, setting a benchmark for encrypted traffic detection models. This framework is particularly relevant to this research as it demonstrates the effectiveness of autoencoder-based models in detecting anomalies in encrypted traffic, providing insights for the development of a similar system for HTTPS traffic.

These studies provide a comprehensive overview of the progress in anomaly detection within encrypted traffic, highlighting the transition from traditional methods to deep learning-based models. While many studies focus on general encrypted traffic or specific protocols such as SSL/TLS or IPsec, this research seeks to extend these findings to the domain of HTTPS traffic, addressing the unique challenges posed by this widely used encryption protocol.

2.5 Anomaly Detection in Encrypted Https Traffic

Currently, HTTPS is the dominant protocol on the Internet, and malicious software exploits HTTPS tunnels to mask their traffic. Encryption, specifically at the TLS handshake stage, allows only partial access to network data for analysis, leaving most of the content hidden, and then increasing the difficulty in identifying malicious activity as attackers exploit HTTPS tunnels for hiding malware traffic. This situation has prompted the need for anomaly detection techniques specifically tailored to the intricacies of HTTPS traffic.

Bingxu Wang et al. (2022) explored various detection methods for HTTPS traffic used over time, categorizing them into four approaches: deep packet inspection, machine learning, behavioural detection, and deep learning. *Deep Packet Inspection* involves analysing TLS handshake fingerprint. For example, He et al. (2014) used TLS fingerprinting and packet length to identify Tor traffic, achieving an FP rate of less than 0.01. However, Tor is dynamic and can easily modify its fingerprint to evade detection. Similarly, Frolov and Wustrow (2019) evaluated cipher suites and extension lists from TLS handshakes to develop client TLS fingerprints, which Anderson and McGrew (2019) expanded by also extracting TLS versions and cipher suites to identify network applications. Althouse et al. (2017) created a fingerprint for the client-server negotiation process and used MD5 hashing to generate a 32-character JA3

fingerprint. Despite these efforts, a critical issue remains: different applications can share identical fingerprint features, reducing the precision of these methods in detecting specific anomalies within web traffic.

Machine Learning approaches, on the other hand, focus on extracting traffic sequences and statistical features to identify traffic using machine learning models. Lashkari et al. (2024) identified 32 time-based flow features and resulted on 92% accuracy using the KNN algorithm. Ibraheem et al. (2022) used an HTTPS dataset with 13 scenarios representing specific types of network attacks to train and evaluate various machine learning models such as Random Forest, SVM, and MLP. Their results showed high precision in identifying abnormal traffic, with Random Forest achieving over 97% accuracy. Liang Wang et al. (2015) focused on flow features extraction for detecting Meek traffic in HTTPS tunnels, achieving 98% accuracy Daniele Ucci et al. (2021) introduce a cybersecurity analytics framework designed to oversee encrypted network traffic and identify features to detect potential attacks and anomalies. This framework integrates machine learning with statistical methods, attaining a 96.6% accuracy rate on malicious datasets and maintaining a false positive rate of around 0.001%. Feghhi and Leith (2016) relied solely on temporal characteristics to detect encrypted HTTPS traffic and achieved promising results. While machine learning methods excel in recall rates, models trained on closed datasets often struggle to perform well in open network environments because of the complexity of novelty in malware and malicious activities.

Behavioural Detection approaches detect tunnel traffic by examining host behaviour. Karagiannis et al. (2005) introduced the BLINC method, which identifies unknown traffic based on host connections. Bingxu Wang et al. (2022) presented the MTBD method, a multi-step detection approach based on multi-dimensional traffic behaviours. With MTBD, a burst detection algorithm first filters out 85% of normal traffic. Then, heterogeneous features are extracted at the flow, host, and packet levels to overcome one-dimensional limitations. Finally, machine learning creates models from these dimensions capable of determining final outcomes through voting. MTBD achieves 98.94% accuracy and 99.08% precision and recall on tested dataset, surpassing many current methods.

Deep Learning approaches reduce feature evaluation complexity by automatically selecting features through training. To address rapid classification and data imbalance issues in encrypted malicious traffic on the Internet, Luo et al. (2021) proposed a classification method based on the DCGAN_1D-CNN model. This model uses generative adversarial networks to

create additional samples at the data level to compensate for the lack of original training samples and sample imbalances. It then uses ID-CNN to train both original and generated samples, accurately classifying encrypted malicious traffic. Results on the CICAndMal2017 dataset achieved an F1 score of 96.55% in binary classification experiments, up to 11.77% higher than using the 1D-CNN model alone. W. Wang et al. (2017) proposed a method using one-dimensional CNN to classify end-to-end encrypted network traffic, directly feeding raw traffic bytes into the neural network without prior feature extraction. He and She (2018) used a deep learning-based SSH traffic classification algorithm to identify applications in tunnel traffic. The result goes up to 95%. Han et al. (2022) employ an unsupervised anomaly detection technique that integrates a three-layer Autoencoder for feature compression, enhancing the model's operational efficiency. They combine this with the traditional K-means algorithm to perform unsupervised classification. This approach can achieve an F1-score of 0.95, which is competitive with supervised learning algorithms.

All these methods represent significant strides in addressing the challenges posed by detecting HTTPS traffic anomalies. Each approach offers unique advantages and challenges, highlighting the ongoing complexity of securing networks against sophisticated threats leveraging encrypted web traffic.

2.6 Challenges in Encrypted Web Traffic Anomaly Detection

The world of technology is in perpetual flux, and so are the challenges related to the security of information systems and the networks through which they communicate and transmit data. The efforts previously mentioned have significantly contributed to improving anomaly detection techniques in encrypted web traffic, but they all have their challenges and limitations.

Traditional methods such as Deep Packet Inspection (DPI) and Intrusion Detection Systems (IDS) have long been relied upon for detecting anomalies, they are increasingly considered inadequate for today’s encrypted web traffic environment. DPI, for instance, relies on reading packet contents, which is not possible in encrypted traffic, while IDS systems are only effective at detecting known anomalies based on signature matching. As noted by Garcia-Teodoro et al. (2009), these conventional systems fail to capture the increasingly complex and encrypted nature of modern web traffic. This critical limitation has prompted the development of advanced machine learning and deep learning techniques tailored to encrypted traffic analysis.

The emergence of machine learning techniques for encrypted traffic detection has shown notable promise. However, these methods depend heavily on the availability of high-quality training data. Althouse et al. (2017) identified the limitations of machine learning, particularly in its inability to adapt to evolving threats. Machine learning models, once trained, often struggle to detect new forms of anomalies that were not present in their training data. This lack of generalization across different datasets poses a challenge in dynamic and open network environments where new attack patterns frequently emerge. Furthermore, the scarcity of labelled data from real-world encrypted traffic hinders the effective training of robust models, as highlighted by Lashkari et al. (2024), who emphasized that data availability is a key bottleneck in applying machine learning to network anomaly detection.

Similarly, deep learning techniques, while more sophisticated and capable of detecting complex patterns, come with their own set of challenges. Han et al. (2022) noted that deep learning models excel in pattern recognition within encrypted traffic but are computationally expensive and often lack transparency in their decision-making processes. These "black box" models do not reveal the underlying features or rationale that lead to anomaly detection decisions, which complicates their interpretability and trustworthiness for network administrators. Moreover, deep learning models require vast amounts of labelled training data, and their computational resource demands make them difficult to deploy in real-time and at scale for many organizations. Luo et al. (2021) found that deep learning models, while powerful, are still limited by the quality of data and the balance of benign versus malicious traffic, and they may not generalize well to new attack scenarios.

Despite these limitations, deep learning remains one of the most promising approaches for detecting anomalies in encrypted web traffic. Continued research in this area seeks to address the existing challenges through novel techniques such as transfer learning, hyperparameter optimization, and hybrid models. For example, Bingxu Wang et al. (2022) demonstrated the potential of combining multiple algorithms to improve detection accuracy and reduce false positives in encrypted traffic. Hybrid approaches, which integrate the strengths of machine learning and deep learning, could potentially overcome some of the limitations associated with each method when applied independently. However, a significant research gap remains in current anomaly detection systems, as they are still unable to effectively analyse and identify malicious activities within encrypted HTTPS traffic without decryption. Existing tools struggle with the complexity of encryption, leaving organizations vulnerable to cyber threats that exploit

encrypted channels to evade detection. This research aims to bridge that gap by developing a deep learning-based system capable of detecting anomalies in encrypted traffic without decryption, thus addressing a critical need in the cybersecurity landscape.

Chapter 3 : Research Methodology

3.1 Introduction

The research methodology provides the structured approach used in this study to address the problem of detecting anomalies in encrypted web traffic using deep learning methods. It serves as a blueprint for systematically collecting, interpreting, and analysing data to answer the research questions, ensuring the reliability and validity of the findings. According to Goundar (2012), a research methodology outlines the procedures and techniques that guide a study, offering a clear framework for acquiring knowledge. In line with the Design Science Research (DSR) methodology, this chapter outlines the phases and methods that were employed to design, develop, and evaluate a deep learning-based anomaly detection system for encrypted web traffic, culminating in the creation of a purposeful artifact.

3.2. Research Design.

This research adopted the Design Science Research (DSR) methodology to develop and validate a deep learning-based anomaly detection system for encrypted HTTPS traffic as illustrated in Figure 3.1. The primary goal was to create an effective model capable of identifying malicious activities without decrypting traffic.

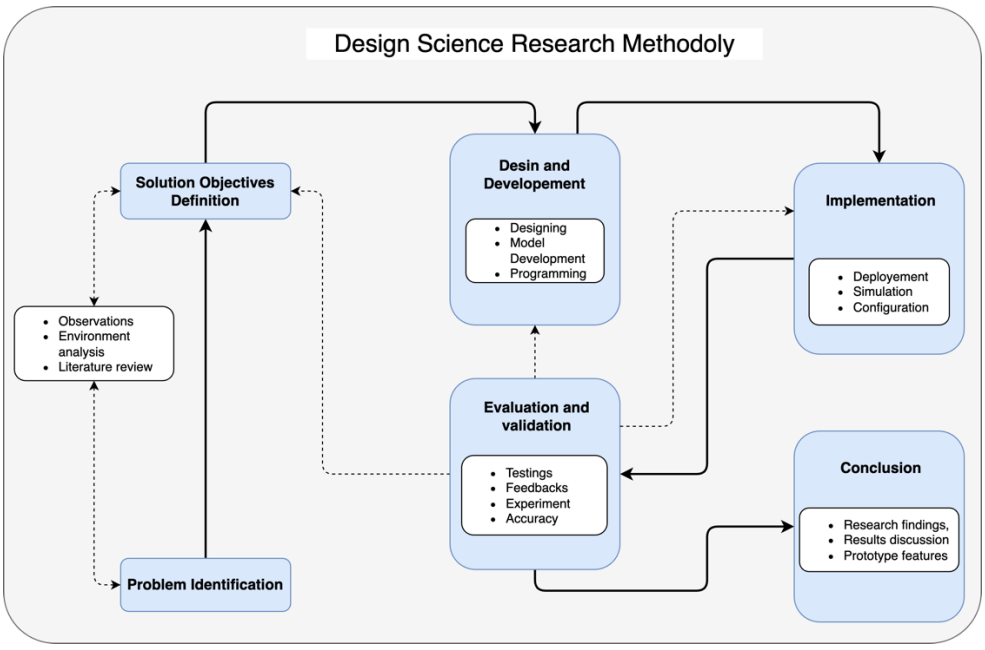


Figure 3.1: Design Science Research (DSR) Methodology

The process began with problem identification, exploring cybersecurity threats in encrypted web traffic, including malware distribution, data exfiltration, phishing, and command-and-

control communications. This involved analysing encrypted traffic patterns and network characteristics to identify indicators of malicious activity.

Next, existing anomaly detection solutions, including Deep Packet Inspection (DPI) and machine/deep learning techniques, were evaluated to identify their limitations and gaps. This assessment informed the design of a novel deep learning model, utilizing Autoencoders (AEs).

The model's development involved iterative data collection, analysis and processing, feature extraction, training, and testing on real encrypted web traffic dataset. Validation was conducted in an offline web environment developed to evaluate the model, assessing performance metrics like detection accuracy, false positive rates, and response times.

The research employed an exploratory and experimental approach, grounded in deep learning and anomaly detection theories. Tools like TensorFlow, NumPy, Matplotlib, Pandas, and seaborn, and scikit-learn were used for model development and data analysis. Real network traffic data will be collected, cleaned, and transformed for training. Model performance was evaluated using accuracy, recall, F1-score, and false positive rates, with cross-validation to ensure generalization.

3.3 Model Development

The Figure 3.2 below illustrate the process use to develop a deep learning model for anomaly detection in encrypted web traffic.

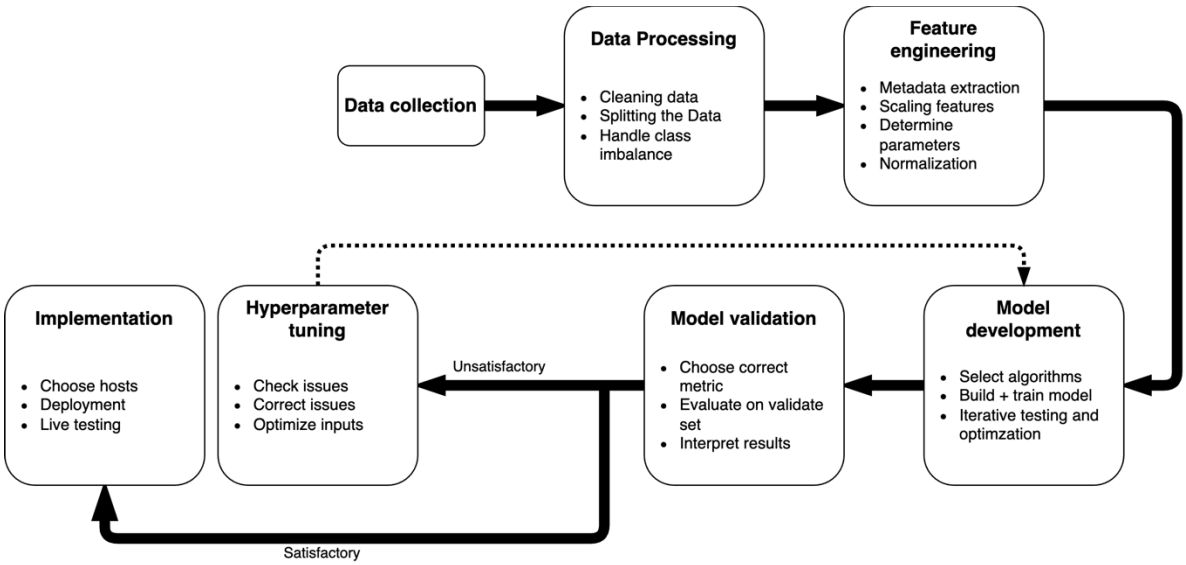


Figure 3.2: Model development process

3.3.1 Data Collection

The model development process begins with data acquisition. For anomaly detection in encrypted web traffic, the dataset must include both normal and abnormal encrypted web traffic. To achieve this, we trained our deep learning model using a public dataset that combines six public datasets collected and released between 2012 and 2020 by Stratosphere Lab and the Canadian Institute for Cybersecurity (CIC) (Z. Wang & Thing, 2022). This dataset provides researchers and cybersecurity professionals with a comprehensive and up-to-date resource for analysing encrypted traffic, which constitutes modern web communications.

3.3.1.1 Data Source

The encrypted traffic dataset utilized in this research was developed by combining six publicly available datasets widely recognized in the cybersecurity research community. These datasets originate from two primary sources: the Malware Capture Facility Project by Stratosphere Lab and several well-known collections published by the Canadian Institute for Cybersecurity (CIC). The final dataset includes both encrypted malicious and legitimate traffic, carefully balanced to ensure an equal representation of each category. The Tables 3.1 and 3.2 represent the composition of the dataset.

Table 3.1: Benign Records of the Dataset

Benign Datasets	**Encrypted Session**
CTU Normal Captures Dataset	79619
CIRA-CIC-DoHBrw-2020 Dataset	105524
CICIDS-2017 Dataset	92975
CICIDS-2012 Dataset	26209
Summary	**304327**

Table 3.2: Malicious Traffic Records in the dataset

Malicious Traffic Type	**Encrypted Session**
Ammyy	14245
Artemis Trojan	10246
Barys	19438
Bunitu Botnet	8060
Bunitu botnet (stripped)	5560
Caphaw or Kazy	24948
Cerber Ransomware	26253
Dridex	6225

HPEMOTET	13736
htbot	10606
Miuref	4634
omQUd	11257
PUA.Taobao	11341
Ransom.Locky	26960
Razy	3207
Sathurbot	1361
TrickBot	8752
Trickster	11644
Trojan.Banker	9296
Trojan.Yakes	1820
trojandownloader	3015
Upatre	1251
Ursnif	10552
Vawtrak	26632
WisdomEyes	24228
Zbot with others	11062
Summary	**306329**

This dataset was curated by selecting raw PCAP/PCAPNG files from the original sources, prioritizing those with sufficient encrypted content. Traffic files with minimal or no encrypted data, or with insufficient packet volume, were excluded. Only relevant encrypted traffic was retained, with protocols unrelated to encrypted web communications—such as ARP and ICMP—removed during preprocessing. Feature extraction was then carried out using a custom Python-based tool that leverages libraries like *dpkt*, *scapy*, and *communityid*, resulting in two CSV datasets: one at the packet level and another at the session level, both connected by a unique session identifier.

In total, 305 traffic features were extracted to support a range of machine learning and deep learning models. The session-level dataset captures the behaviour of entire encrypted communication flows, while the packet-level dataset allows for finer-grained analysis. Additionally, to reduce data skew and control variability, session truncation was applied—limiting sessions to a maximum of 15 packets. This ensured consistent structure across sessions and optimized model training.

3.3.1.2 Collection and Anonymization Methods

The collection and preprocessing approach focused on building a reliable dataset for encrypted malicious traffic detection, while maintaining data integrity and ensuring user privacy. The process began with the acquisition of raw network traces from six publicly available datasets,

followed by a rigorous selection phase to retain only relevant encrypted communications. Packets associated with non-encrypted or auxiliary protocols, such as ARP and ICMP, were filtered out to focus the dataset on encrypted traffic behaviours.

After filtering, a custom Python-based feature extraction tool was used to transform raw traffic into structured data. This tool, built using libraries like dpkt, scapy, and communityid, extracted 305 features per flow or packet, depending on the dataset type. The extracted data was then exported into two separate but related CSV files: one at the packet level and one at the session level, with both files linked via a unique hash identifier. Additionally, to normalize the dataset and control session length variability, a truncation method was applied to limit the maximum number of packets per session to 15.

To preserve user anonymity and protect against data re-identification risks, comprehensive anonymization was carried out. Although the traffic itself is encrypted, metadata still poses potential privacy concerns. As such, all personally identifiable information and network-level identifiers were removed or masked. MAC addresses and other hardware fingerprints were excluded, and all source IP addresses were replaced with a static private address (10.0.2.105), while destination IPs were uniformly set to a placeholder value (195.113.214.236). This step ensured that no traceable information could be linked back to original users or systems.

The formats of the collected and anonymized data, covering both packet-based-dataset and session-based-dataset, are illustrated in Figure 3.3 and Figure 3.4 respectively.

```
'Time_cost': 3195812.18365,
'Flag_of_packets': 24,
'Traffic_sequence': 360828843,
'Payload_ratio': 0.6875,
'Length_of_IP_packets': 128,
'Length_of_TCP_payload': 88,
'Length_of_TCP_packet_header': 20,
'Length_of_IP_packet_header': 20,
'TCP_windows_size_value': 64240,
'Length_of_TCP_segment(packet)': 108,
'Time_difference_between_packets_per_session': 0.0002270000986754,
'Change_values_of_TCP_windows_length_per_session': 0,
'Interval_of_arrival_time_of_forward_traffic': 0.0002270000986754,
'Interval_of_arrival_time_of_backward_traffic': 0.0,
'Time_to_live': 128,
'Ratio_to_previous_packets_in_each_session': 3.2,
'Source_port': 51561,
'Destination_port': 443,
'source_IP_address': '10.0.2.105',
'Destination_IP_address': '195.113.214.251',
'inter_arrival_time_of_forward_traffic_enc': 0.0,
'inter_arrival_time_of_backward_traffic_enc': 0.0,
'ratio_to_previous_packet_enc': 0.0,
'label': 1,
'unique_link_mark': 6043858606213562857
```


Figure 3.3: An example of packet record from the packet-based dataset.

Below is a table outlining the meaning of each feature present within the packet-based-dataset:

Table 3.3: Packet based dataset features

Key	Description
Time_cost	Total time (in microseconds) consumed for the packet transmission.
Flag_of_packets	TCP flags captured for the packet (e.g., ACK, SYN).
Traffic_sequence	Unique sequence number identifying traffic flow order.

Payload_ratio	Ratio of payload size to total packet size.
Length_of_IP_packets	Total length of the IP packet.
Length_of_TCP_payload	Length of TCP payload within the packet.
Length_of_TCP_packet_header	Size of the TCP header.
Length_of_IP_packet_header	Size of the IP header.
TCP_windows_size_value	TCP window size value indicating buffer size.
Length_of_TCP_segment(packet)	Total length of TCP segment (header + payload).
Time_difference_between_packets_per_session	Time gap between this packet and the previous one in the session.
Change_values_of_TCP_windows_length_per_session	Changes in TCP window size within the session.
Interval_of_arrival_time_of_forward_traffic	Time interval since the last forward-direction packet.
Interval_of_arrival_time_of_backward_traffic	Time interval since the last backward-direction packet.
Time_to_live	Time-To-Live (TTL) value for the packet.
Ratio_to_previous_packets_in_each_session	Ratio of current packet size to previous packets in the session.
Source_port	Source port number of the packet.
Destination_port	Destination port number (443 usually denotes HTTPS).
source_IP_address	Anonymized IP address of the source.
Destination_IP_address	Anonymized IP address of the destination.
inter_arrival_time_of_forward_traffic_enc	Encrypted traffic inter-arrival time (forward direction).
inter_arrival_time_of_backward_traffic_enc	Encrypted traffic inter-arrival time (backward direction).

ratio_to_previous_packet_enc	Ratio of this encrypted packet size to the previous one.
label	Label indicating whether the traffic is malicious (1) or benign (0).
unique_link_mark	Unique identifier linking this packet to a session.

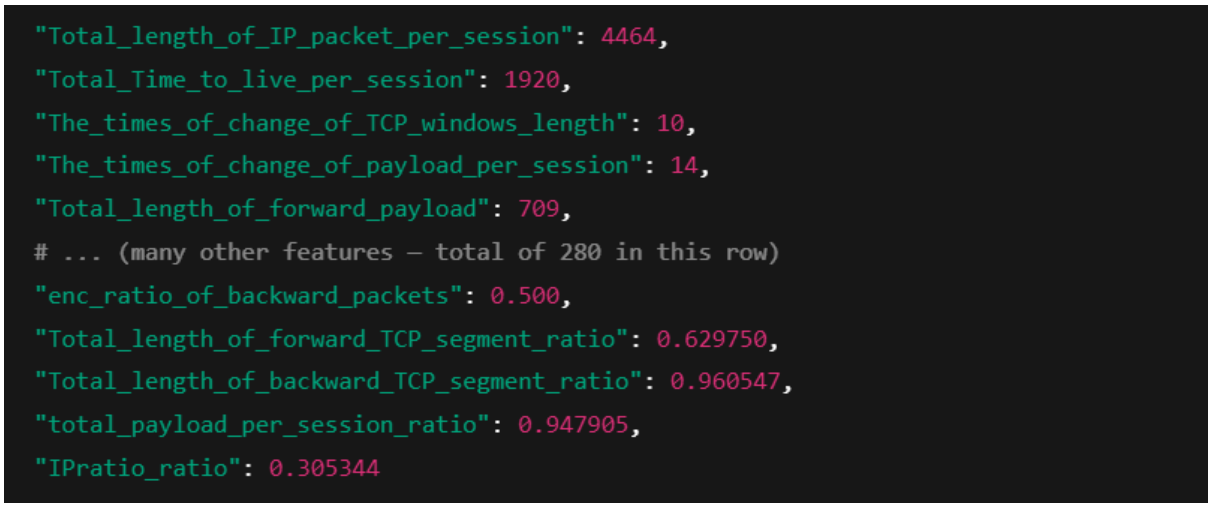


Figure 3.4: An example of a packet from the network traffic data

Here are some of the session-based features from the dataset along with what they represent in the context of encrypted traffic analysis:

Table 3.4: Some session_based_dataset features

Feature Name	**Description**
Total_length_of_IP_packet_per_session	Total size (in bytes) of all IP packets in the session.
Total_Time_to_live_per_session	Sum of the Time-To-Live (TTL) values for all packets in the session.
The_times_of_change_of_TCP_windows_length	Number of times the TCP window size changed during the session—can indicate session dynamics or congestion control behaviour.

The_times_of_change_of_payload_per_session	Number of payload length changes throughout the session, possibly reflecting interactive or non-uniform communication.
Total_length_of_forward_payload	Total size of the payload sent in the forward direction (e.g., client to server).
Total_length_of_backward_payload	Total size of the payload sent in the reverse direction (e.g., server to client).
Total_length_of_forward_IP_header	Sum of IP header sizes for all packets sent forward.
Total_length_of_backward_IP_header	Sum of IP header sizes for all packets sent backward.
Total_length_of_forward_TCP_header	Sum of TCP header sizes in the forward direction.
Total_length_of_backward_TCP_header	Sum of TCP header sizes in the backward direction.
Total_length_of_forward_TCP_segment(packet)	Total size of TCP segments (header + payload) sent forward.
Total_length_of_backward_TCP_segment(packet)	Same as above but for the reverse direction.
flow duration	Total duration of the session in microseconds.
No_of_forward_packets	Total number of packets sent from client to server.
No_of_backward_packets	Total number of packets sent from server to client.
Bytes_From_Clients(IPpacket)	Total number of bytes sent from clients, including headers.
Bytes_From_Servers(IPpacket)	Total number of bytes sent from servers, including headers.

mean_Length_of_IP_packets	Average IP packet size over the entire session.
median_Length_of_IP_packets	Median IP packet size in the session.
max_Length_of_IP_packets	Maximum observed IP packet size.

3.3.2 Data Processing

The preprocessing phase commenced with the cleaning of both the packet-based and session-based datasets. This involved the identification and removal of rows containing missing or NaN values to ensure data consistency. Features such as source and destination IP addresses were excluded, as they held no relevance for the training objectives and could potentially introduce noise or bias. Additionally, the unique_link_mark, which served only to link corresponding entries across the two datasets, was removed from the training data to prevent the model from learning identifiers rather than behavioural patterns.

All remaining features were validated for consistency in data type and semantic relevance. The resulting datasets contained only those attributes deemed essential for model training and anomaly detection in encrypted web traffic.

Following data cleaning, exploratory data analysis was conducted to assess the structure and distribution of the datasets. Correlation matrices were generated to identify dependencies and multicollinearity among features, informing subsequent feature selection. The class distribution between normal and anomalous traffic was analysed across the training and testing sets, for both packet-level and session-level datasets, to confirm dataset balance.

Subsequently, the data was partitioned into training, validation, and test subsets to support model development and evaluation. All numeric features were normalized using the StandardScaler to standardize the range and variance of feature values. This scaling process ensured that high-magnitude features did not disproportionately influence the model during the training phase.

3.3.3 Feature Engineering

Feature engineering prioritized metadata and behavioural characteristics over content-based attributes due to the encrypted nature of the traffic. The retained features included metrics such as byte counts, packet counts, session durations, inter-arrival times, and protocol-specific

indicators. These attributes were selected for their capacity to capture structural and temporal patterns relevant to anomaly detection.

The engineered features were preserved following the cleaning and scaling processes. All features included in the final datasets contributed to the representation of encrypted communication behaviours, forming a reliable input set for the development of anomaly detection models.

3.3.4 Model Development

To detect anomalies in encrypted web traffic, two deep learning models were developed—one for the packet-based dataset and another for the session-based dataset. Both models were designed as binary classifiers based on fully connected neural networks, leveraging supervised learning due to the availability of labelled data indicating normal and anomalous traffic.

The input features for each model were obtained from the pre-processed and scaled datasets. The packet-based model utilized flow-level statistics such as byte counts, packet counts, and timing-related attributes, while the session-based model incorporated aggregated behavioural features extracted from session-level flows. In both cases, features unrelated to behavioural analysis, such as IP addresses and identifiers, were excluded to ensure the models focused solely on encrypted traffic characteristics.

Each model was implemented using the TensorFlow/Keras framework and followed a similar architecture: a feedforward neural network consisting of three hidden layers with decreasing units (128, 64, 32), all activated using the ReLU function. The final output layer used a sigmoid activation function to perform binary classification. This structure enabled the models to capture non-linear relationships between features and accurately distinguish between normal and anomalous patterns.

The models were compiled using the Adam optimizer with a learning rate of 0.0001 and binary cross-entropy as the loss function, suitable for the classification objective. The training process was conducted over 300 epochs with a batch size of 256. Data was split in an 80:10:10 ratio, with 80% used for training, 10% for validation, and 10% for testing. The training and validation datasets were used to optimize the learning process, while the test set was reserved for final evaluation.

Both models were trained independently and are intended to be combined within a unified detection framework. This two-model design allows the system to leverage both granular (packet-level) and aggregated (session-level) insights to enhance anomaly detection accuracy in encrypted web traffic.

The two Figures 3.5 and Figure 3.6 represent the structure used to train the two models

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	2,816
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

Total params: 13,185 (51.50 KB)
Trainable params: 13,185 (51.50 KB)
Non-trainable params: 0 (0.00 B)

Figure 3.5: Packet-based training model structure

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	35,840
dense_1 (Dense)	(None, 64)	8,256
dense_2 (Dense)	(None, 32)	2,080
dense_3 (Dense)	(None, 1)	33

Total params: 46,209 (180.50 KB)
Trainable params: 46,209 (180.50 KB)
Non-trainable params: 0 (0.00 B)

Figure 3.6: Session-based training model structure

3.3.5 Model Validation

Model validation was conducted to assess the predictive performance, generalization capacity, and robustness of the binary classification models designed for anomaly detection in encrypted web traffic. Two distinct models were evaluated: a packet-based model, which processes individual network packets, and a session-based model, which aggregates traffic at the session level. The validation process employed both standard and advanced supervised learning metrics to quantify model performance, including accuracy, precision, recall, F1 score, Area Under the Receiver Operating Characteristic Curve (AUC), and generalization gap.

Furthermore, confusion matrices were analysed to provide insight into classification behaviour, particularly regarding false positive and false negative rates.

Both models were trained using 80% of the labelled dataset, with 10% allocated for validation and 10% for final testing. All data subsets underwent identical preprocessing, including feature scaling and dimensionality reduction via feature selection, ensuring consistency across model inputs.

3.3.5.1 Packet-Based Model Evaluation

The packet-based model was first validated on the designated test dataset after training and hyperparameter tuning. It achieved an accuracy of 98.63% on the validation set and 98.64% on the test set, indicating a high degree of consistency and effective generalization. The binary cross-entropy loss function returned values of 0.0364 and 0.0367 for the validation and test sets, respectively. These low loss values indicate that the model's predicted probability distributions were very close to the actual labels.

Further, on the test set, the model achieved a precision of 98.67%, meaning that 98.67% of all packets classified as anomalies were indeed anomalous. The recall score was 98.36%, showing that the model successfully detected 98.36% of all actual anomalous packets. The F1 score, which balances precision and recall, was calculated to be 98.51%, confirming a strong trade-off between false positives and false negatives. In addition, the model yielded an AUC score of 99.93%, indicating excellent discriminatory power between normal and anomalous traffic. The generalization gap, defined as the absolute difference between validation and test accuracy, was only 0.0004, demonstrating strong generalization and negligible overfitting. The table 3.5 shows all the considered metrics.

Table 3.5 Packet-Based Model Performance Metrics

Metric	Validation Set	Test Set
Accuracy	98.63%	98.64%
Loss	0.0364	0.0367
Precision	–	98.67%
Recall	–	98.36%
F1 Score	–	98.51%
AUC Score	–	99.93%

Generalization Gap	–	0.0004

The confusion matrix, as illustrated in Figure 3.7 further supports these performance metrics. It showed that 99,07% of normal packets were correctly classified as normal (true negatives), while 0,93% normal packets were misclassified as anomalous (false positives). On the anomaly side, the model correctly detected 98.36% of anomalous packets (true positives), and 1.64% of anomalies were misclassified as normal (false negatives). These results underscore the model's capacity to maintain a low false alarm rate while still capturing nearly all true anomalies.

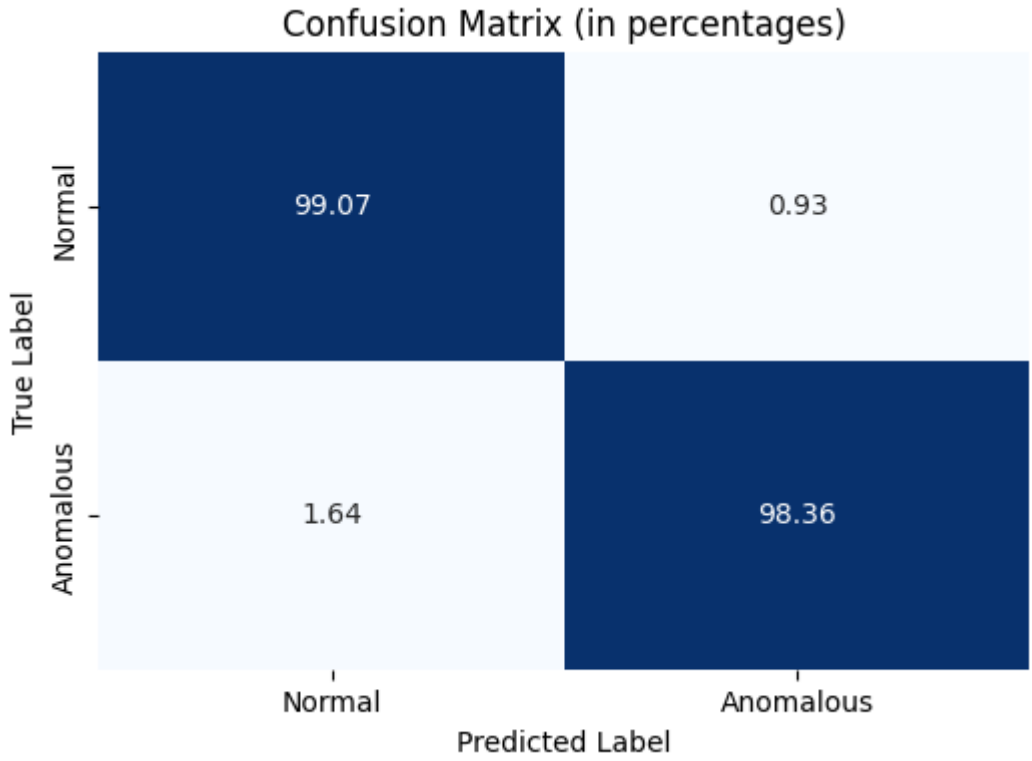


Figure 3.7 Confusion Matrix for the Packet-Based Model

The Figure 3.8 and the learning curves, as illustrated in Figure 3.11, confirm stable convergence of both accuracy and loss, with no signs of overfitting.

```
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 41s 2ms/step - accuracy: 0.9879 - loss: 0.0310 - val_accuracy: 0.9857 - val_loss: 0.0353
Epoch 297/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 42s 2ms/step - accuracy: 0.9878 - loss: 0.0311 - val_accuracy: 0.9875 - val_loss: 0.0319
Epoch 298/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 43s 2ms/step - accuracy: 0.9879 - loss: 0.0312 - val_accuracy: 0.9873 - val_loss: 0.0319
Epoch 299/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 43s 2ms/step - accuracy: 0.9879 - loss: 0.0309 - val_accuracy: 0.9897 - val_loss: 0.0282
Epoch 300/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 43s 2ms/step - accuracy: 0.9879 - loss: 0.0310 - val_accuracy: 0.9863 - val_loss: 0.0364

Evaluating on Validation Set...
Validation Loss: 0.0364
Validation Accuracy: 0.9863 (98.63%)

Evaluating on Test Set...
Test Loss: 0.0367
Test Accuracy: 0.9864 (98.64%)
```

Figure 3.8: Validation Performance Metrics of the Packet-Based Model

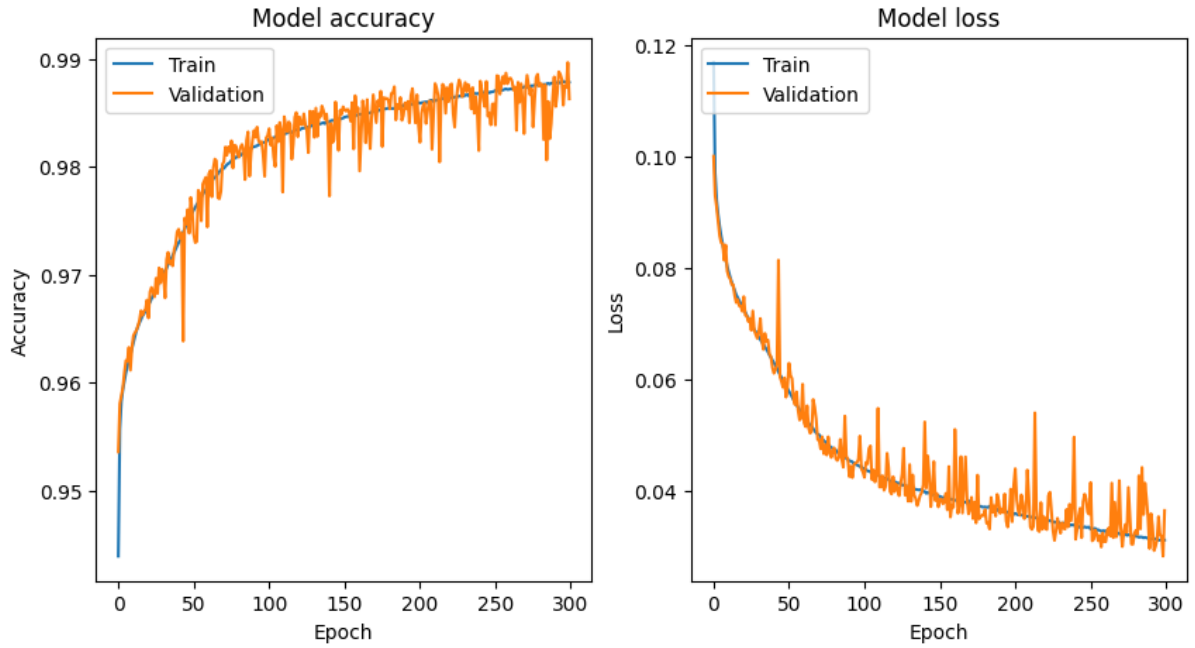


Figure 3.9: Validation Accuracy and Loss Curve for the Packet-Based Model

3.3.5.2 Session-Based Model Evaluation

The session-based model demonstrated even stronger validation performance as presented in table 3.6. It achieved an accuracy of 99.67% on the validation set and 99.66% on the test set, confirming near-perfect classification accuracy at the session level. The validation and test loss values were 0.0267 and 0.0318, respectively, both lower than those of the packet-based model. These figures reflect better optimization and convergence performance.

On the test set, the session-based model yielded a precision of 99.75%, signifying that almost every predicted anomaly was indeed anomalous. The recall was 99.64%, meaning the model was able to correctly detect nearly all anomalous sessions. The F1 score reached 99.69%, which demonstrates outstanding balance between precision and recall. Additionally, the AUC score

of 99.97% confirmed the model's ability to accurately differentiate between normal and anomalous sessions. The generalization gap was calculated as 0.0026, slightly higher than that of the packet-based model, but still within acceptable limits, especially considering the higher classification complexity at the session level.

Table 3.6 Session-Based Model Validation Metrics

Metric	Validation Set	Test Set
Accuracy	99.67%	99.66%
Loss	0.0267	0.0318
Precision	–	99.75%
Recall	–	99.64%
F1 Score	–	99.69%
AUC Score	–	99.97%
Generalization Gap	–	0.0026

The confusion matrix for the session-based model revealed that 99.75% normal sessions were correctly predicted (true negatives), with only 0.25% normal sessions misclassified as anomalous (false positives). It also correctly classified 99.74% anomalous sessions (true positives), while misclassifying only 0.36% anomalous sessions as normal (false negatives). The figure 3.10 illustrates the model's extremely low rate of both false positives and false negatives.

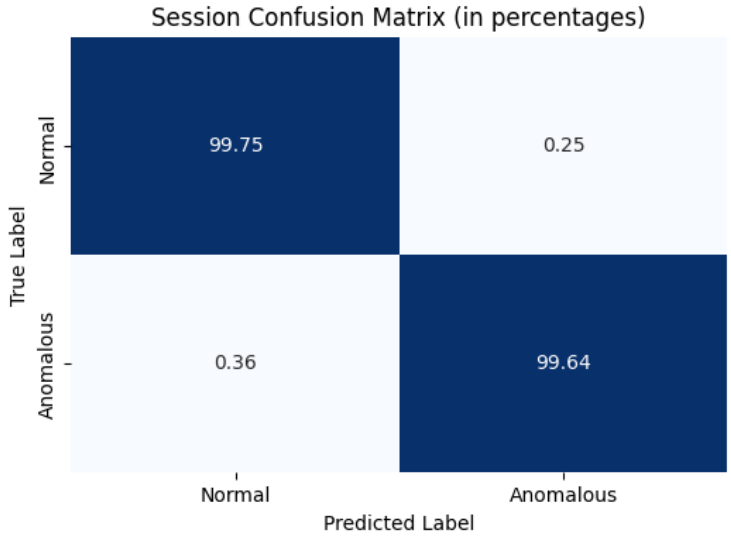


Figure 3.10 Session Confusion Matrix

Figures 3.11 and 3.12 show the validation metrics and learning curves for validation accuracy and loss, both of which reveal stable training dynamics and optimal convergence.

```
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 14s 8ms/step - accuracy: 0.9995 - loss: 0.0014 - val_accuracy: 0.9963 - val_loss: 0.0332
Epoch 298/300
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 19s 7ms/step - accuracy: 0.9996 - loss: 0.0012 - val_accuracy: 0.9966 - val_loss: 0.0292
Epoch 299/300
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.9996 - loss: 0.0011 - val_accuracy: 0.9966 - val_loss: 0.0275
Epoch 300/300
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.9995 - loss: 0.0013 - val_accuracy: 0.9967 - val_loss: 0.0267

Evaluating on Validation Set...
Validation Loss: 0.0267
Validation Accuracy: 0.9967 (99.67%)

Evaluating on Test Set...
Test Loss: 0.0318
Test Accuracy: 0.9966 (99.66%)
```

Figure 3.11 Validation Performance Metrics of the Session-Based Model

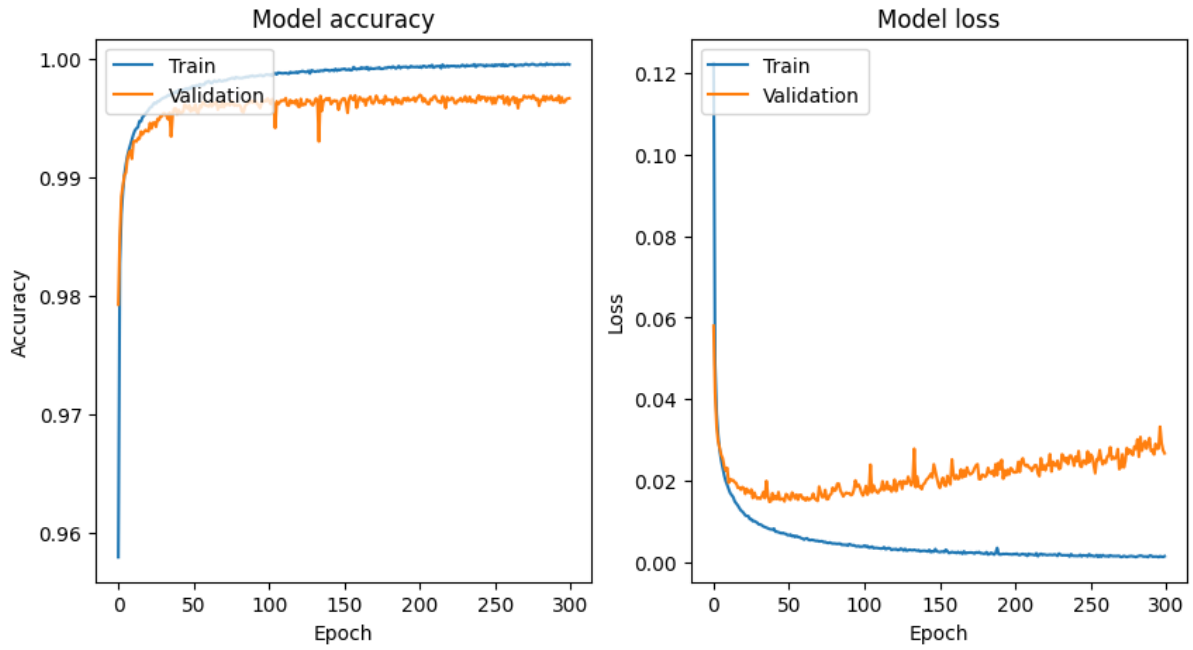


Figure 3.12: Validation Accuracy and Loss Curve for the Session-Based Model

3.3.5.3 Comparative Analysis

To provide a direct comparison between the two models, Table 3.7 summarizes their key performance metrics. While both models exhibited outstanding results, the session-based model slightly outperformed the packet-based model in every key area, including accuracy, precision, recall, F1 score, and AUC. The only exception was the generalization gap, which was marginally larger in the session-based model, likely due to its more complex representation of temporal and contextual relationships.

Table 3.7 Comparative Performance of Packet-Based vs. Session-Based Models

Metric	Packet-Based Model	Session-Based Model
Accuracy	98.64%	99.66%
Precision	98.67%	99.75%
Recall	98.36%	99.64%
F1 Score	98.51%	99.69%
AUC Score	99.93%	99.97%
Generalization Gap	0.0004	0.0026
Loss (Test Set)	0.0367	0.0318

From this comparison, it is evident that session-level modelling introduces a valuable contextual understanding of encrypted traffic behaviour that enhances detection capability. However, the packet-based model remains highly effective and is computationally more efficient, which may be a critical factor in real-time deployment scenarios.

3.4 Model Implementation

The ultimate objective of this research is to deploy a comprehensive anomaly detection system capable of operating in real-time environments and integrating with a Security Information and Event Management (SIEM) system. Although the current scope is limited to offline testing, the design and development of the models have been carried out with deployment readiness in mind.

To evaluate the combined effectiveness of the two trained models—packet-based and session-based—a unified inference module was developed. This module integrates both models into a single decision framework, enabling joint analysis of test data and improving detection robustness by leveraging both granular and aggregated traffic perspectives.

For demonstration and testing purposes, the integrated module was implemented in a Python-based web application. This application facilitates model inference, result visualization, and interaction with the test datasets. While this implementation does not yet operate in a live network environment, it serves as a functional prototype and proof of concept for future integration into production systems.

This stage of implementation is positioned as a preparatory step toward a full deployment pipeline, where real-time traffic data can be processed and analysed continuously, and alerts can be forwarded to a SIEM for monitoring and response. The details of the developed web application, system integration logic, and performance observations are discussed in Chapter 5, which presents the implementation and results in greater depth.

3.5 Result Interpretation

Due to limitations in time, infrastructure, and the complexity involved in capturing and processing real-time encrypted traffic, the implemented system was not deployed in a live production environment. Instead, evaluation was conducted using a labelled HTTPS dataset specifically selected to simulate real-world encrypted web traffic, including both normal and malicious sessions.

The purpose of this evaluation was to assess the effectiveness of the integrated detection system in identifying anomalous behaviours based on the combined output of the packet-based and session-based models. The test results demonstrated that the system was capable of accurately detecting various forms of anomalous traffic patterns, including simulated scanning activity, abnormal request volumes, and irregular client behaviour.

Although the models were not applied to live traffic, the use of a structured test dataset allowed for reliable assessment of system performance in a controlled environment. The results confirmed that the models are capable of generalizing beyond the training data and effectively distinguishing between normal and anomalous encrypted web traffic without relying on payload inspection.

A detailed presentation of the evaluation results, including visual performance metrics and detection outputs, is provided in Chapter 5, which covers the implementation and result discussion.

3.6 Use of Research Findings

The findings of this research are intended to contribute to both academic inquiry and practical advancements in the field of cybersecurity, particularly in the detection of anomalies within encrypted web traffic using deep learning techniques. Academically, this study provides a structured methodology for processing and analysing encrypted traffic using labelled public

datasets. It also demonstrates the effectiveness of combining packet-based and session-based deep learning models for anomaly detection, serving as a foundation for future research in this domain.

The preprocessing techniques, model development strategies, and evaluation procedures described in this work will be documented and made available to the research community through scientific publications, technical presentations, and open-source repositories. This promotes transparency, reproducibility, and collaborative improvement of encrypted traffic analysis methodologies.

From a practical perspective, the research offers valuable insights into building detection systems that can function without decrypting traffic content—an increasingly important consideration in privacy-preserving cybersecurity monitoring. The developed models, though not deployed in real-time environments during this study, are structured for future integration into operational frameworks such as intrusion detection systems (IDS), Security Information and Event Management (SIEM) platforms, or other threat detection tools.

These outcomes may also inform the efforts of cybersecurity professionals and policy-makers by highlighting detection strategies that respect encryption while maintaining threat visibility. In future research, the implementation will be extended to support real-time encrypted traffic analysis and system integration, contributing to more robust, privacy-aware security infrastructures.

3.7. Ethical Considerations

Ethical considerations have been central to this research, particularly due to the sensitive nature of network traffic data. To ensure strict adherence to data privacy standards, no real-world traffic data was captured or processed by the researcher. Instead, publicly available and fully anonymized datasets were used, specifically the Encrypted Traffic Feature Dataset for Machine Learning and Deep Learning-based Encrypted Traffic Analysis, which consolidates six well-established public sources. These datasets were anonymized prior to publication by their original custodians, ensuring that no personally identifiable information (PII) or device-level fingerprints are retained.

The use of anonymized and synthetic traffic ensures that this study complies with data protection and ethical research standards. All experimental procedures were carried out in

isolated, offline environments without impact on operational systems, further reducing the risk of unintended consequences or data exposure.

Transparency and accountability have been maintained through detailed documentation of data processing workflows, model training decisions, and evaluation methods. No attempt has been made to reverse anonymization, infer identity, or link traffic to individuals or organizations. Additionally, secure storage practices were observed for all datasets and experiment outputs throughout the research lifecycle.

As data privacy and ethical standards continue to evolve in the context of cybersecurity, this research remains aligned with emerging best practices and will serve as a responsible model for future work in the secure and ethical development of AI-based anomaly detection systems.

Chapter 4 : System Design and Architecture

This chapter presents the detailed design and architecture of the anomaly detection system developed to identify anomalies in encrypted web traffic—specifically HTTPS—using deep learning techniques. The system is designed with modularity and scalability in mind, allowing it to process and analyse traffic at both the packet and session levels. Leveraging supervised learning models trained on labelled encrypted traffic data, the system can identify abnormal patterns without decrypting the payload content.

The architecture is intended to simulate real-time operation, in which encrypted traffic is analysed as it is ingested. While this study does not include a live deployment due to resource constraints, the system is structured for seamless transition into a real-time detection environment. The design emphasizes accuracy, efficiency, and adaptability for future integration with monitoring tools such as Security Information and Event Management (SIEM) platforms.

4.1 Overall System Architecture.

The proposed system architecture for anomaly detection in encrypted web traffic is illustrated in Figure 4.1.

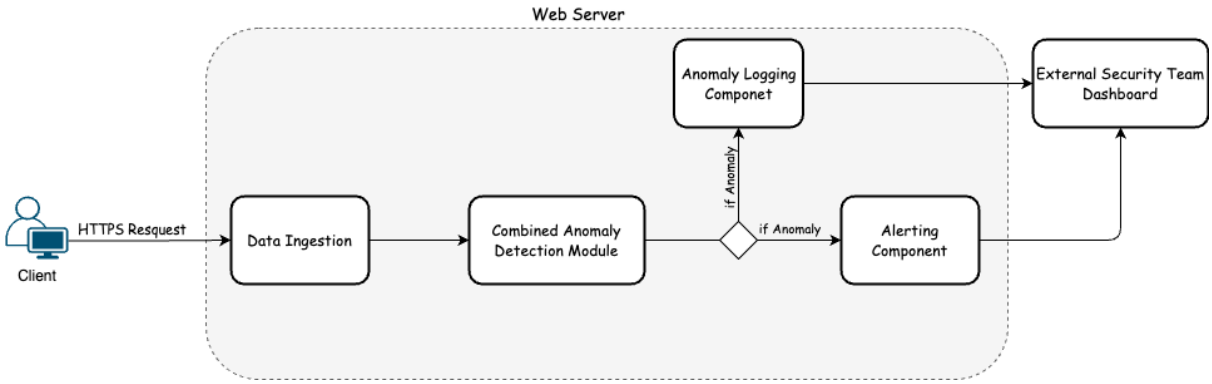


Figure 4.1: Anomaly Detection System Architecture

The system begins with the passive collection of HTTPS traffic, typically performed via a Switch Port Analyser (SPAN) or mirror port configured on a network switch. In a real-time setting, this traffic would be mirrored directly from the network environment; however, for the purpose of this study, a labelled HTTPS dataset was used to simulate the data stream.

Captured traffic is passed through the Data Ingestion Layer, which handles data loading and feeds it to the subsequent processing modules. This system works solely with network flow

data, divided into two categories: packet-based traffic and session-based traffic. These represent different granularities of encrypted communication and allow the system to learn both low-level and aggregated behavioural patterns.

Next, the Traffic Processing Component extracts and prepares features required for classification. This includes preprocessing tasks such as data cleaning, feature scaling, transformation, and encoding, as described in Chapter 3. Two separate data streams are maintained: one for packet-level features and another for session-level features.

Following processing, the feature sets are passed to two independent, pre-trained deep learning models, specialized through training on either packet-based or session-based features respectively, which then each predict whether the observed traffic instance indicates normal or anomalous encrypted behaviour. These predictions are then passed to the Decision Aggregation Module, which combines the outputs from both models. At this level, a voting fusion strategy is be applied here to produce a unified decision, enhancing detection accuracy and reducing false positives.

Once an anomaly is detected, the system forwards the event to two components: the Anomaly Logging Component, which stores detection records with associated metadata (e.g., timestamp, classification confidence, feature vector), and the Alerting Component, which is responsible for triggering notifications or alerts. In a real-world deployment, these alerts would be integrated into a SIEM platform for centralized security monitoring.

The entire system is designed to function passively, ensuring that the HTTPS traffic remains untouched during the capture and analysis process. This architecture emphasizes detection only, aligning with the primary objective of this study. Response mechanisms such as traffic blocking or connection termination fall outside the current scope but are identified as areas for future enhancement.

4.2 Anomaly Detection Module Design

The core component of the anomaly detection system is the Anomaly Detection Module, illustrated in Figure 4-2. This module integrates two independently trained supervised deep learning models—one based on packet-level features and the other on session-level features—to provide a robust and multi-layered detection mechanism for encrypted web traffic.

Both models operate on pre-processed network traffic, derived from the same HTTPS dataset but structured at different granularities. The packet-based model analyses traffic at a lower level, focusing on individual packet characteristics and short-term patterns. In contrast, the session-based model evaluates aggregated metrics over entire sessions, capturing long-term communication behaviours.

The process begins with incoming traffic being parsed and separated into packet-level and session-level formats. Each dataset undergoes a specific preprocessing pipeline, including cleaning, feature selection, and normalization, as described in Chapter 3. The two prepared datasets are then passed into their respective trained models:

i. The Packet-Based Model outputs a binary classification indicating whether the analysed packet sequence is normal or anomalous.
ii. The Session-Based Model performs the same operation on a broader view of the traffic session.

Both models produce independent predictions, which are then forwarded to the Combined Decision Unit. This unit aggregates the outputs using a voting decision strategy. If either model classifies an instance as anomalous, or if the combined confidence score exceeds a predefined threshold, the instance is flagged as an anomaly.

The final decision, along with metadata such as timestamps, classification confidence, and key traffic features, is logged for further review. While the current architecture does not implement response actions such as blocking or mitigation, it is designed to support future integration with real-time defence mechanisms, including firewalls, SIEM platforms, or intrusion prevention systems.

This design ensures high detection accuracy by leveraging complementary strengths of both packet-level precision and session-level context, enhancing the system's capacity to identify sophisticated or obfuscated threats within encrypted web traffic. The architecture and the flow sequence of the combined module are represented by the Figure 4.2 and Table 4.1, respectively.

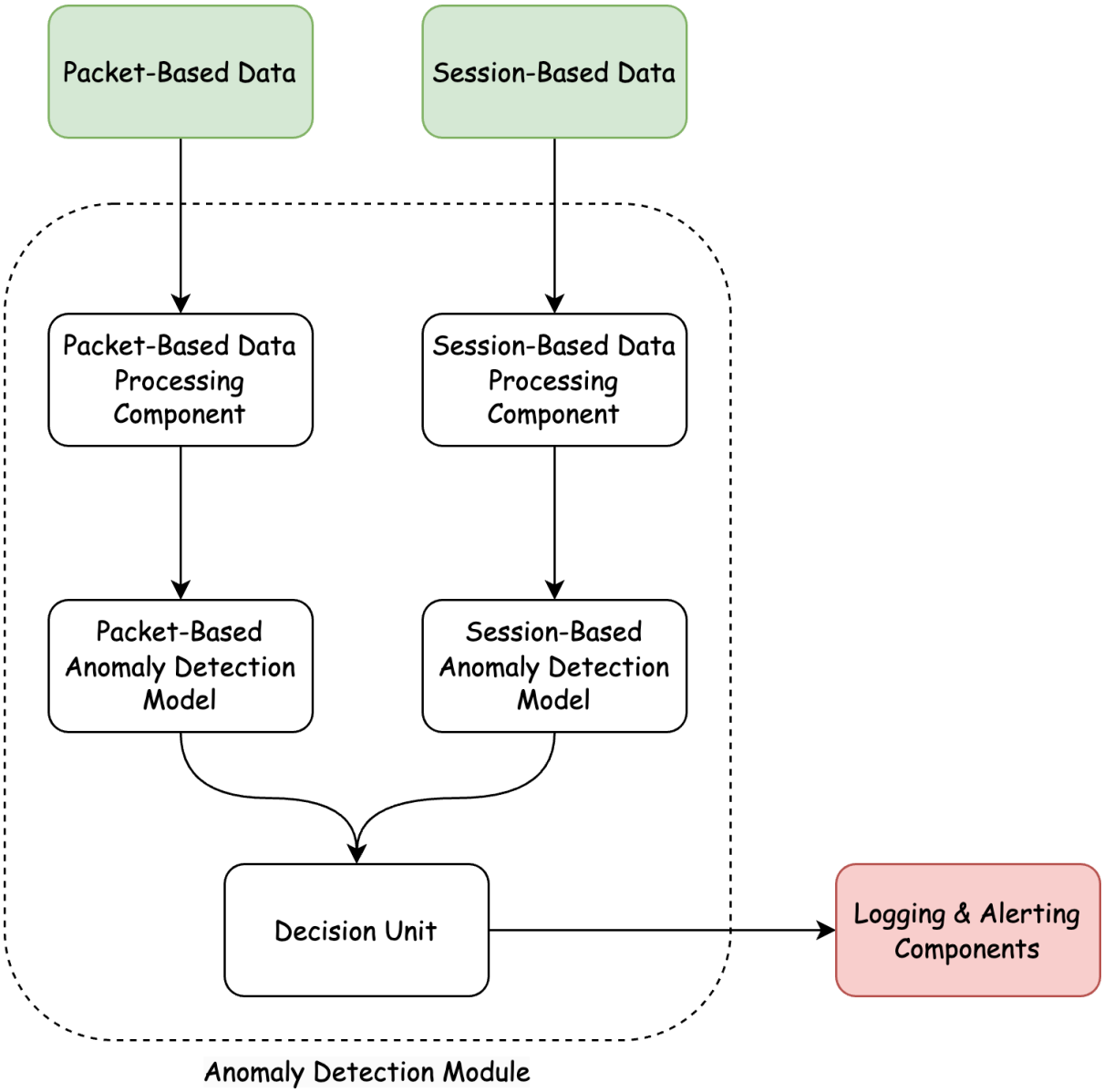


Figure 4.2: Anomaly Detection Module Diagram

Table 4.1: Combined Detection Module Flow Sequence

Time	Component	Action	Data Source
T1	Packet-Based Preprocessor	Receives and preprocesses packet-level traffic	Captured Network Traffic
T1	Session-Based Preprocessor	Receives and preprocesses session-level traffic	Aggregated Network Sessions

T2	Packet-Based Model	Receives and Performs classification on processed packet features	Scaled Packet-Based Features
T2	Session-Based Model	Receives and Performs classification on processed session features	Scaled Session-Based Features
T3	Model Decision Unit	Receives classification results from both models	Binary Output (Packet, Session)
T4	Model Decision Unit	Aggregates predictions (e.g., voting or weighted rule)	Classification Scores
T5	Model Decision Unit	Compares against final anomaly threshold	Final Decision (Anomaly/Normal)

4.3 Detailed Anomaly Design

The system design in this context refers to the functional layout of core components, their specific roles, and the interactions between them to perform supervised anomaly detection on encrypted web traffic. The architecture is structured to facilitate efficient data ingestion, preprocessing, classification, and alerting—ensuring seamless operation in a simulated real-time environment.

The overall process begins with the passive capture of HTTPS traffic. This traffic is separated and formatted into two distinct views: packet-level data and session-level data. These formats represent different analytical perspectives—fine-grained and aggregated—and are processed independently to extract features relevant to the behaviour of encrypted flows.

The Data Ingestion Layer receives the captured HTTPS traffic and forwards it to the Anomaly Detection Module, which houses two trained supervised classification models: i) the Packet-Based Model, optimized for identifying short-term traffic anomalies, and ii) the Session-Based Model, tailored to detect broader, session-level deviations.

Each data stream undergoes dedicated preprocessing, including cleaning, transformation, and normalization, before being fed into its corresponding model. Each model then outputs a binary classification indicating whether the input instance is normal or anomalous.

These results are passed to a Decision Unit, which combines both outputs using a voting rule. If either model flags an anomaly or if the aggregated score surpasses a defined threshold, the system marks the instance as anomalous.

Detected anomalies are logged by the Anomaly Logging Component, storing metadata such as timestamp, classification output, and relevant traffic features. Simultaneously, the Alerting Component can issue notifications to a connected monitoring system or SIEM.

Although the system currently operates in an offline evaluation mode, its modular architecture is designed to support real-time operation and future enhancements such as automated responses (e.g., blocking suspicious IPs or redirecting malicious sessions). We illustrate the full System Design in Figure 4.3 and the Sequence Flow in Table 4.2.

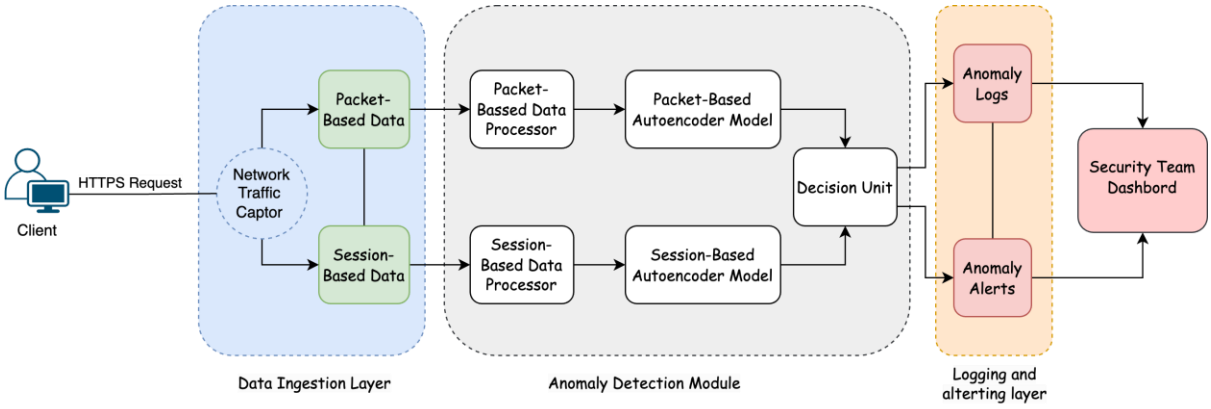


Figure 4.3: System Flow Sequence Diagram

Table 4.2: System Flow Sequence Table

Time	Component	Action	Data
T0	Client	Sends an HTTPS request to the server	HTTPS Packet
T1	HTTPS Traffic Capture	Passively captures HTTPS traffic	Raw HTTPS Traffic
T2	Data Ingestion Layer	Separates Data (Packet-Based Data and Session-Based Data) and forwards raw traffic to the anomaly detection module	Raw HTTPS Traffic
T3	Packet-Based Preprocessor	Processes packet-level data and extracts features	Processed Packet-Level Data

T3	Session-Based Preprocessor	Processes session-level data and extracts features	Processed Session-Level Data
T4	Packet-Based Model	Performs classification on packet-level features	Anomaly Label (Packet-Based)
T4	Session-Based Model	Performs classification on session-level features	Anomaly Label (Session-Based)
T5	Decision Unit	Aggregates model outputs and applies decision logic	Combined Decision (Anomaly/Normal)
T6	Anomaly Logging Component	Logs details of detected anomalies	Anomaly Metadata
T6	Alerting Component	Sends alert notification to security monitoring platform (if enabled)	Alert Notification

4.4. Conclusion

This chapter has presented the detailed design and architecture of the proposed anomaly detection system for encrypted web traffic. The system leverages a dual-model architecture, utilizing both packet-level and session-level classifiers trained through supervised deep learning techniques. These models operate in parallel to analyse encrypted HTTPS traffic and collaboratively identify anomalous behaviour.

The methodology involves using structured, labelled datasets and direct classification outputs, resulting in improved precision and clearer model interpretability. The architecture supports modularity, scalability, and future integration into real-time security environments.

The next chapter provides an overview of the system's implementation, including the web-based testing interface, and discusses the results obtained from evaluating the models using the test dataset.

Chapter 5 Implementation and Discussion

This chapter presents the implementation of the anomaly detection system developed for identifying anomalies in encrypted web traffic using deep learning techniques. It also discusses the results obtained during model testing, the challenges faced during implementation, and the implications of the findings. Although a real-time deployment was not realized due to resource and infrastructure constraints, the trained models were evaluated using a structured HTTPS dataset to simulate realistic conditions. The implementation serves as a functional prototype or a proof of concept, validating the feasibility of the proposed approach and laying the foundation for future deployment.

5.1 Implementation Environment

To simulate practical deployment, a prototype system was developed to test the two trained deep learning models—packet-based and session-based. The system was implemented in a Python-based environment using TensorFlow and Scikit-learn libraries. The models were integrated into a standalone module that accepts pre-processed data and performs real-time inference on test samples.

Rather than deploying the system in a live network environment, publicly available encrypted traffic datasets were used for evaluation. These datasets were derived from PCAP files that had been processed and feature-engineered to match the packet- and session-level input formats used during model training. This offline setup ensured a controlled and repeatable testing process while approximating real-world encrypted traffic behaviour.

Below, the interface of the offline setup implementation as represented in the Figure 5.1.

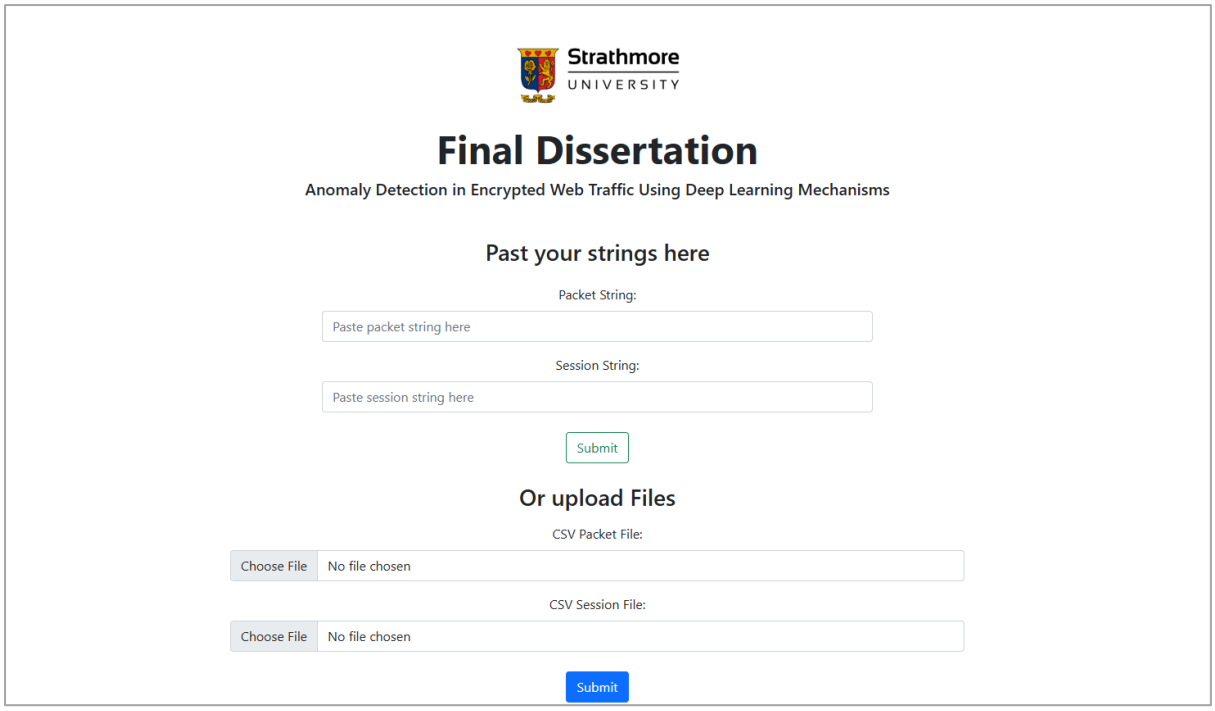


Figure 5.1: Anomaly Detection Test Interface

5.2 Test Dataset and Attack Simulation

To evaluate the performance of the anomaly detection system under realistic conditions, a structured and labelled dataset was used. This dataset included both benign HTTPS traffic and malicious encrypted traffic generated by a variety of malware families. The use of this dataset enabled the simulation of real-world encrypted traffic scenarios, allowing for comprehensive testing of the trained models without requiring access to live or uncontrolled network environments.

The malicious traffic in the dataset covered a wide range of malware types, including banking trojans, botnets, ransomware, and downloader variants. Specifically, the dataset contained encrypted web communications associated with the following malware families: *Ammyy, Artemis Trojan, Barys, Bunitu Botnet (including a stripped variant), Caphaw or Kazy, Cerber Ransomware, Dridex, HPEMOTET, htbot, Miuref, omQUd, PUA.Taobao, Ransom.Locky, Razy, Sathurbot, TrickBot, Trickster, Trojan.Banker, Trojan.Yakes, trojandownloader, Upatre, Ursnif, Vawtrak, WisdomEyes, and Zbot with others*. These samples represent a diverse spectrum of attack vectors and behaviours commonly encountered in modern cyber threats, particularly in encrypted communications.

Each malicious instance was labelled and associated with encrypted traffic sessions or flows generated during malware execution. These traffic patterns exhibit characteristics such as abnormal connection frequencies, irregular port or protocol usage, short-lived connections, suspicious TLS handshake behaviours (e.g., missing or spoofed Server Name Indication), and unusual payload sizes or timing. Importantly, because the traffic was encrypted, the detection models had no visibility into content-level features and relied solely on metadata and behavioural indicators.

In addition to malicious data, the dataset included a substantial volume of benign encrypted traffic, representing typical user behaviour in enterprise or internet environments. This normal traffic included HTTPS connections for browsing, updates, and application usage. The presence of clean data was essential for testing the system's ability to avoid false positives, ensuring that legitimate encrypted sessions would not be incorrectly classified as anomalous.

Together, the benign and malicious samples allowed for balanced and realistic model evaluation. The diversity and labelling of traffic types ensured that the models could be rigorously tested for both detection accuracy and robustness, confirming their suitability for practical deployment in environments where encryption is pervasive and content inspection is infeasible.

5.4 Detection Results and Observations

The evaluation of the anomaly detection system was conducted by testing the trained models on a held-out portion of the encrypted HTTPS dataset. Both the packet-based and session-based models were tested independently on unseen data to assess their ability to generalize and accurately identify anomalous traffic. The performance was evaluated using supervised learning metrics, specifically accuracy and binary cross-entropy loss, with results demonstrating strong model effectiveness.

The packet-based model, which analyses fine-grained, low-level features extracted from individual flows, achieved a test accuracy of 98.64% and a loss of 0.0367 as represented in Figure 5.2. This performance indicates that the model was highly effective in detecting short-term or bursty malicious behaviour, such as port scanning or repeated connection attempts. Its ability to identify anomalies at this level is particularly important in scenarios where attackers use low and slow techniques that evade session-level detection.

```
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 41s 2ms/step - accuracy: 0.9879 - loss: 0.0310 - val_accuracy: 0.9857 - val_loss: 0.0353
Epoch 297/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 42s 2ms/step - accuracy: 0.9878 - loss: 0.0311 - val_accuracy: 0.9875 - val_loss: 0.0319
Epoch 298/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 43s 2ms/step - accuracy: 0.9879 - loss: 0.0312 - val_accuracy: 0.9873 - val_loss: 0.0319
Epoch 299/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 43s 2ms/step - accuracy: 0.9879 - loss: 0.0309 - val_accuracy: 0.9897 - val_loss: 0.0282
Epoch 300/300
17326/17326 ━━━━━━━━━━━━━━━━━━━━ 43s 2ms/step - accuracy: 0.9879 - loss: 0.0310 - val_accuracy: 0.9863 - val_loss: 0.0364

Evaluating on Validation Set...
Validation Loss: 0.0364
Validation Accuracy: 0.9863 (98.63%)

Evaluating on Test Set...
Test Loss: 0.0367
Test Accuracy: 0.9864 (98.64%)
```

Figure 5.2: Packet-Based Model Test Results (Accuracy and Loss)

The session-based model, which operates on aggregated features over entire encrypted sessions, yielded even higher performance, achieving a test accuracy of 99.66% and a loss of 0.0318 as shown in Figure 5.3. This model proved especially effective in identifying anomalies associated with sustained or complex attacks such as brute-force login attempts, malware command-and-control communications, and behaviourally distinct TLS handshake patterns. Its success can be attributed to its broader temporal context and ability to capture traffic trends beyond the scope of individual flows.

```
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 14s 8ms/step - accuracy: 0.9995 - loss: 0.0014 - val_accuracy: 0.9963 - val_loss: 0.0332
Epoch 298/300
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 19s 7ms/step - accuracy: 0.9996 - loss: 0.0012 - val_accuracy: 0.9966 - val_loss: 0.0292
Epoch 299/300
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 10s 6ms/step - accuracy: 0.9996 - loss: 0.0011 - val_accuracy: 0.9966 - val_loss: 0.0275
Epoch 300/300
1507/1507 ━━━━━━━━━━━━━━━━━━━━ 13s 8ms/step - accuracy: 0.9995 - loss: 0.0013 - val_accuracy: 0.9967 - val_loss: 0.0267

Evaluating on Validation Set...
Validation Loss: 0.0267
Validation Accuracy: 0.9967 (99.67%)

Evaluating on Test Set...
Test Loss: 0.0318
Test Accuracy: 0.9966 (99.66%)
```

Figure 5.3: Session-Based Model Test Results (Accuracy and Loss)

To combine the strengths of both models, a decision aggregation strategy was implemented based on majority voting, wherein an input instance is classified as anomalous if at least one of the two models flags it as such. Formally, the decision rule can be represented as:

$$D_{combined}(x) = \begin{cases} 1 \; if \; D_{packet}(x) = 1 \; or \; D_{session}(x) = 1 \\ 0 \; otherwise \end{cases}$$

Where:

- $D_{packet}(x)$ is the binary prediction from the packet-based model for instance x.

- $D_{session}(x)$ is the binary prediction from the session-based model for the same instance.
- $D_{combined}(x)$ is the final decision of the combined module.

Given the known accuracy values for both models, and assuming statistical independence between their predictions (for estimation purposes), the expected accuracy of the combined decision module using a voting strategy was approximated by:

$$Accuracy_{combined} = 1 - (1 - Accuracy_{packet})(1 - Accuracy_{session})$$

Substituting the known values:

$$Accuracy_{combined} = 1 - (1 - 0.9864)(1 - 0.9966) = 1 - (0.0136\ x\ 0.0034)$$
$$= 1 - 0.00004624 = 99.995\%$$

This theoretical upper bound suggests that combining both models results in a significant enhancement in detection reliability. While this assumes ideal independence and no correlation in error patterns, it provides a conservative estimate of the potential benefit of a multi-model system.

In addition to this theoretical accuracy, a broader set of combined evaluation metrics was also derived to reflect practical performance more holistically. The combined precision—computed as the average of the precision scores from both models—was 99.21%, while the combined recall stood at 99.00%, indicating that nearly all actual anomalies were detected and few false alarms were generated. The combined F1 score was calculated as 99.10%, demonstrating a well-balanced trade-off between sensitivity and specificity.

The generalization gap for the combined model was calculated by summing the gaps from the individual models' training and test accuracies. This yielded a value of 0.0030, which remains low and indicates that the combined system maintains strong generalization capability even when aggregating decisions across different levels of abstraction.

The practical benefits of the combined model were clearly demonstrated during evaluation. Several cases of evasive malicious traffic—missed individually by one model—were successfully detected when the decisions were aggregated. This confirms that the packet- and session-level perspectives are complementary rather than redundant, and their fusion creates a more resilient detection layer that can withstand a wider range of adversarial behaviors.

These results support the core design principle of the system: leveraging multi-level analysis to strengthen detection performance without relying on payload inspection. The outcome demonstrates the feasibility of applying deep learning to encrypted traffic anomaly detection and sets the stage for future deployment in real-time monitoring environments. The visual representation in Figure 5.4 confirms that the combined model performs well in distinguishing between normal and anomalous encrypted web traffic. The green marker positioned near the top of the graph indicates near-perfect accuracy, while the red marker near the bottom confirms that the model maintained very low prediction error during testing.

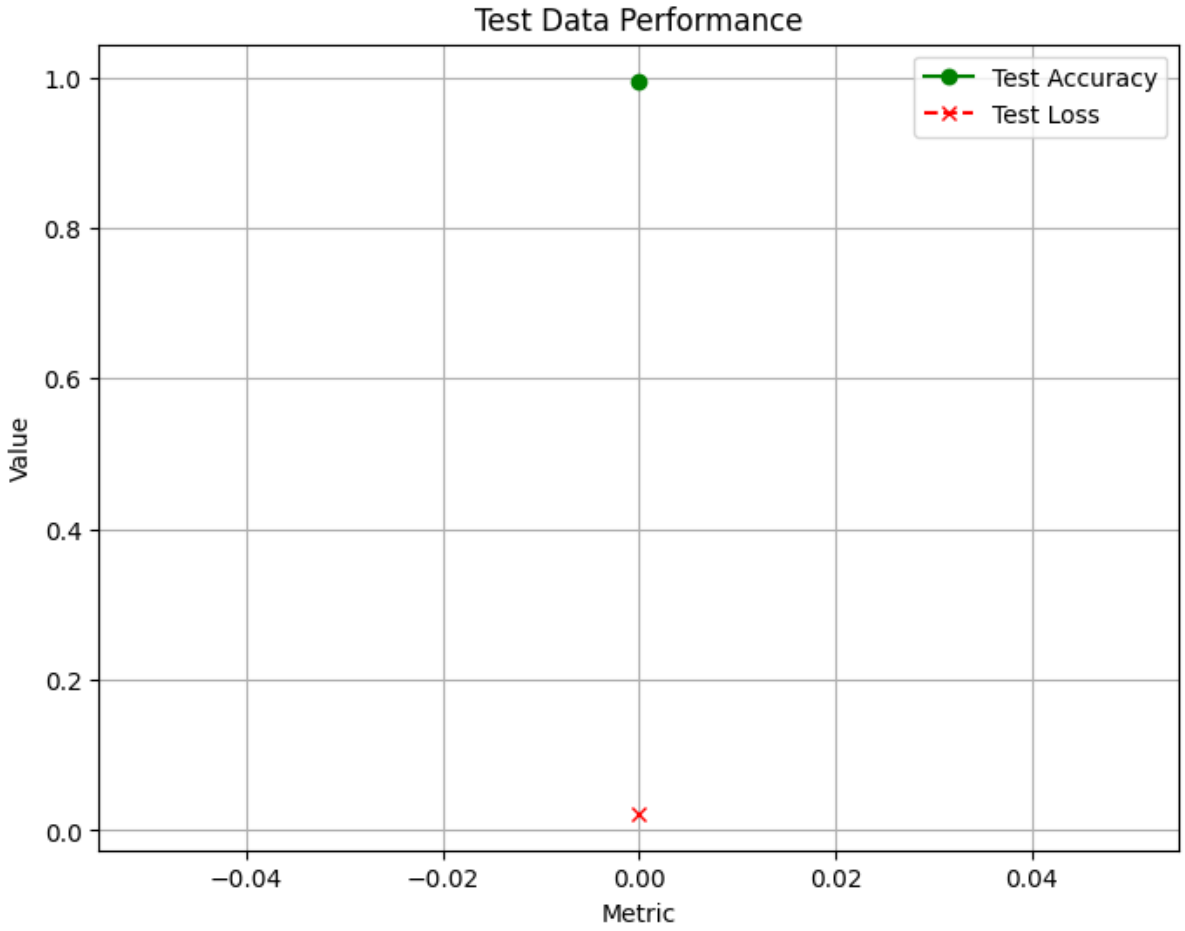


Figure 5.4: Visual Representation of the Combined Model Performance

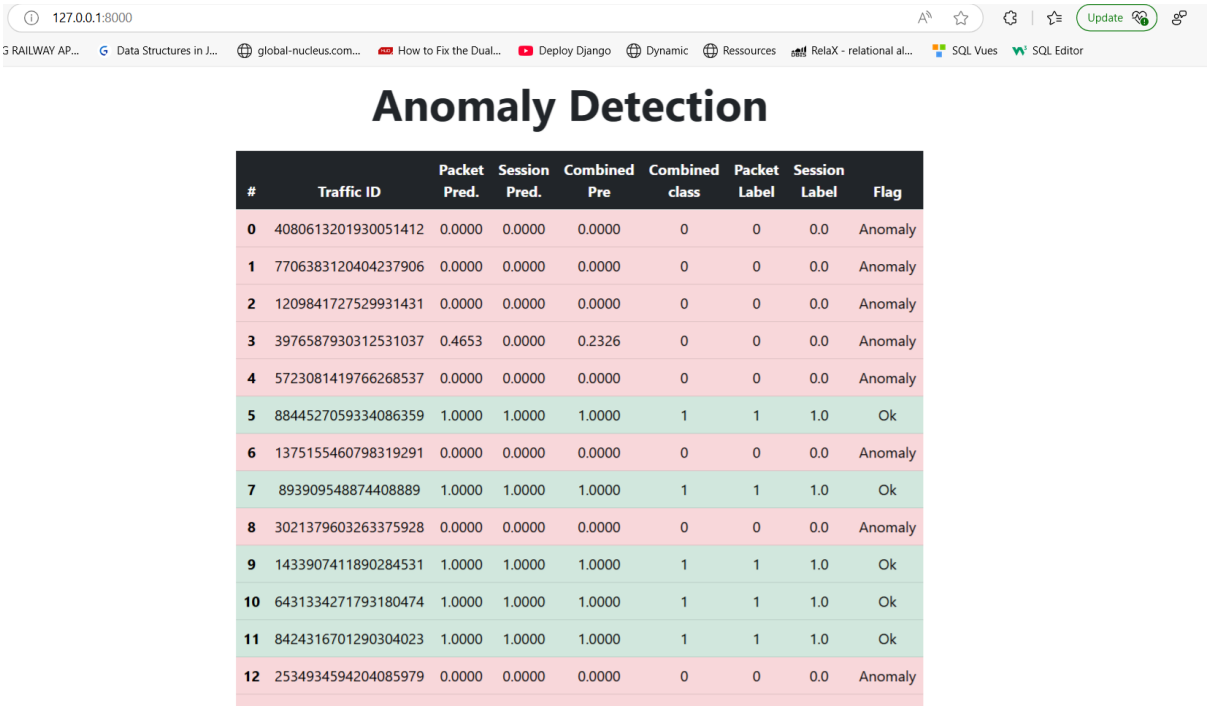
5.5 Discussion of Results and Practical Impact

The results obtained from testing the anomaly detection system confirm the feasibility and practical value of applying supervised deep learning techniques to encrypted web traffic. The packet-based model achieved an accuracy of 98.64%, while the session-based model reached 99.66%, demonstrating their respective effectiveness in identifying a wide range of anomalies across different granularities. These strong performances were attained without decrypting any

content, relying entirely on encrypted traffic metadata—a fundamental advantage in today's privacy-conscious digital ecosystem. As web services increasingly adopt end-to-end encryption by default, the inability to inspect payloads directly renders traditional intrusion detection methods ineffective. This system circumvents that limitation by learning from structural and behavioural characteristics such as flow duration, packet count, inter-arrival times, TLS handshake features, and traffic volume patterns—features that remain visible even within fully encrypted sessions.

The dual-model architecture emerges as a core strength of this approach. The packet-based model is well-suited for detecting rapid, short-lived anomalies like port scans or burst-based DoS attacks, while the session-based model provides a broader perspective, capturing contextual anomalies that span extended traffic flows. Their combination through a voting-based decision strategy significantly enhances the detection landscape, enabling the system to capitalize on the unique advantages of both models. Theoretical analysis based on the assumption of independence between model predictions suggests a combined system accuracy of approximately 99.995%. While this figure is mathematically derived and should be interpreted as an upper bound rather than an absolute performance guarantee, it still highlights the robustness and complementarity of the models working in tandem.

The system's combined metrics further reinforce this strength. Precision and recall values for the integrated model were calculated at approximately 99.21% and 99.00%, respectively, indicating that the system not only detects threats effectively but also does so with minimal false positives. The F1 score of 99.10% confirms a well-balanced trade-off between detection sensitivity and specificity. Furthermore, the generalization gap—the difference in performance between the training and testing phases—remained low at 0.0030, underscoring the models' ability to generalize well to unseen data and reducing concerns about overfitting. These high-level results are particularly impressive given the complexity of encrypted traffic and the absence of decrypted payload information. The Figure 5.5 below shows an illustration of the combined model capability in detection.



#	Traffic ID	Packet Pred.	Session Pred.	Combined Pre	Combined class	Packet Label	Session Label	Flag
0	4080613201930051412	0.0000	0.0000	0.0000	0	0	0.0	Anomaly
1	7706383120404237906	0.0000	0.0000	0.0000	0	0	0.0	Anomaly
2	1209841727529931431	0.0000	0.0000	0.0000	0	0	0.0	Anomaly
3	3976587930312531037	0.4653	0.0000	0.2326	0	0	0.0	Anomaly
4	5723081419766268537	0.0000	0.0000	0.0000	0	0	0.0	Anomaly
5	8844527059334086359	1.0000	1.0000	1.0000	1	1	1.0	Ok
6	1375155460798319291	0.0000	0.0000	0.0000	0	0	0.0	Anomaly
7	893909548874408889	1.0000	1.0000	1.0000	1	1	1.0	Ok
8	3021379603263375928	0.0000	0.0000	0.0000	0	0	0.0	Anomaly
9	1433907411890284531	1.0000	1.0000	1.0000	1	1	1.0	Ok
10	6431334271793180474	1.0000	1.0000	1.0000	1	1	1.0	Ok
11	8424316701290304023	1.0000	1.0000	1.0000	1	1	1.0	Ok
12	2534934594204085979	0.0000	0.0000	0.0000	0	0	0.0	Anomaly

Figure 5.5: Output of Anomaly Detection Testing interface

Handling false positives (FP) and false negatives (FN) was a key consideration throughout the system's design and evaluation. Given the high stakes in cybersecurity—where a false positive may overwhelm security analysts and a false negative may allow a threat to go undetected—the combination of the packet-based and session-based models served as a deliberate strategy to minimize these risks. Each model, while effective on its own, exhibited slight variability in its classification boundary due to differences in temporal scope and feature sensitivity. By integrating both models through a voting-based decision mechanism, the system achieved a form of decision-level redundancy, enabling one model to offset potential misclassifications by the other. Although no detection system can entirely eliminate FP and FN occurrences, the resulting performance metrics—an accuracy of 99.995%, a precision of 99.21%, and a recall of 99.00%—suggest that the combined system reduces these errors to a practically negligible level within the bounds of supervised deep learning. This demonstrates that thoughtful model fusion can be a viable mitigation approach, particularly in privacy-preserving environments where payload inspection is not an option.

From a practical perspective, the system is built on a modular and extensible architecture. Each component, ranging from data ingestion and feature extraction to classification and decision fusion, operates independently and can be scaled or replaced without affecting the overall functionality. This makes the system suitable for integration into cloud-native environments,

Security Operations Centre workflows, and Security Information and Event Management (SIEM) systems. Additionally, the system aligns well with regulatory frameworks such as GDPR (General Data Protection Regulation) by enabling threat detection without infringing on data privacy. It offers a balanced solution that supports security operations while upholding ethical data handling practices.

Nevertheless, certain limitations must be acknowledged. The combined accuracy figure of 99.995%, while indicative of strong performance, assumes that the two models produce independent predictions. In real-world environments, this assumption may not always hold true, especially given the potential correlation between packet- and session-level traffic features. As such, the combined metric may slightly overestimate performance and should be considered an idealized projection. Moreover, the method used to aggregate precision, recall, and F1 scores across models—by averaging—provides a general indication of performance but may obscure specific weaknesses. For instance, if one model contributes disproportionately to anomaly detection in a particular class of attacks, this averaging could mask uneven detection capabilities. A more nuanced analysis could be useful in identifying such edge cases and optimizing model roles accordingly.

Another point worth noting is that while the system demonstrated consistent results on a structured dataset of encrypted traffic, its behaviour under high-throughput, real-time network conditions remains to be fully validated. Testing was conducted in a simulated environment, and although the system showed strong accuracy and efficiency, its scalability and performance in live enterprise or ISP-grade deployments require further investigation. Also, while the session-based model provides valuable context over time, neither model incorporates long-term behavioural profiling or user-specific baselining. This lack of historical context may limit the detection of slow-moving or low-and-slow attacks. Future research could explore integrating temporal modelling techniques, such as LSTM or Transformer-based architectures, to enhance state awareness and adaptive learning.

5.6 Implementation Tools and Technologies

The implementation of the anomaly detection system was carried out using a set of open-source tools and libraries, with a focus on flexibility, reproducibility, and ease of integration. The Table 5.1 below list all the tools and technologies used for the implementation of the solution.

Table 5.1: List of Implementation Tools and Technologies

Tool / Technology	Purpose	Description
Python	Core programming language	Used for all stages of the project: data analysis, feature engineering, model development, and integration.
Pandas / NumPy	Data processing	Handled data cleaning, manipulation, transformation, and numerical computations.
Scikit-learn	Preprocessing and evaluation	Used for splitting datasets, scaling features, and evaluating model performance.
TensorFlow / Keras	Deep learning model development	Enabled construction, training, and testing of the packet-based and session-based classification models.
Matplotlib / Seaborn	Visualization	Used to generate plots such as accuracy/loss curves, confusion matrices, and feature distributions.
Jupyter Notebook	Interactive development	Provided a flexible environment for writing, testing, and documenting code during the research phase.
Google Colab	Cloud-based model training	Assisted in running computationally intensive tasks when local hardware was insufficient.
Django	Web framework for deployment	Used to build a prototype web app that integrates the trained models for testing and visualization of results.
VS Code	Development environment	Served as the primary IDE for writing and organizing the Django-based application and Python scripts.

5.7 Implementation Limitations and Callenges

Despite the strong performance of the models during testing, several limitations affected the current implementation. The most significant challenge was the inability to implement real-

time detection, primarily due to limited computational resources and the absence of specialized hardware for high-throughput traffic capture and processing. The proposed system is designed to analyse encrypted traffic in real time—ideally identifying anomalies before the traffic reaches and is processed by the target server. However, live detection, which is critical for proactive security and early decision-making, could not be realized at this stage.

Implementing real-time traffic monitoring involves several technical complexities, including the deployment of SPAN ports or passive network taps, packet-to-flow conversion, and real-time feature extraction. These steps must be integrated with a high-speed inference pipeline capable of low-latency classification. Achieving this would require a robust infrastructure, such as GPU-accelerated processing units, parallel I/O systems, and optimized data pipelines. This challenge—real-time encrypted traffic anomaly detection with minimal latency—remains a key priority for future work.

Another limitation lies in the absence of a streaming data pipeline. Currently, the system operates in batch mode, where data is captured, pre-processed, and evaluated offline. While suitable for initial validation, this approach does not replicate the dynamic conditions of a live production environment, where continuous data flow, memory management, and inference efficiency must be addressed.

The system also lacks integration with real-time alerting mechanisms. Anomalies are logged locally, but no alerts are generated or forwarded to a SIEM, SOC dashboard, or firewall. This restricts the system's usefulness in operational cybersecurity settings. A complete deployment would require automated response options, including configurable alert thresholds, log forwarding, and potential mitigation actions (e.g., blocking IPs or flagging suspicious sessions for review).

An additional limitation concerns the interpretability of detected anomalies. While the models—whether used independently or in combination—are effective in detecting the presence of anomalous traffic, they do not provide insights into the type or nature of the anomaly. That is, the models produce a binary classification (normal or anomalous) without indicating which malware family, attack type, or behavioural category triggered the alert. As a result, each flagged instance must undergo manual or external analysis to determine the underlying cause or identity of the anomaly. This limitation restricts the system's utility for

rapid incident response or forensic categorization, and highlights the need for future enhancements in explainability and classification depth.

Despite these limitations, the implementation successfully validated the core concept of using deep learning to detect anomalies in encrypted web traffic based solely on observable metadata. The findings establish a solid groundwork for extending the system into a real-time, fully integrated, and operationally deployable security solution.

5.8. Proposed Strategies for Mitigating Limitations

To mitigate the current implementation limitations, several strategic improvements can be considered to advance the system toward real-time, operational deployment. Addressing the challenge of real-time detection could involve upgrading the computational infrastructure by incorporating graphics processing unit accelerated processors or specialized artificial intelligence inference hardware such as tensor processing units or field-programmable gate arrays. These resources may support the high-throughput and low-latency demands of encrypted traffic analysis. Complementary to hardware improvements, leveraging high-performance packet capture technologies (for example, Data Plane Development Kit or Packet Filter Ring) might ensure efficient and lossless data ingestion. Real-time feature extraction and model inference could be integrated into a scalable stream processing framework, potentially using platforms such as Apache Flink or Apache Kafka Streams, to enable continuous, low-latency processing. Architecting the system as containerized microservices orchestrated via Kubernetes may further improve scalability and fault tolerance.

Transitioning from batch-mode to a fully streaming data pipeline is essential to replicate dynamic, real-world network environments. This may involve implementing continuous flow construction, feature extraction, and normalization components supported by reliable message queues such as Apache Kafka or Apache Pulsar to buffer and stream data effectively. Model inference can be embedded within this pipeline using dedicated model serving solutions (for example, TensorFlow Serving or NVIDIA Triton Inference Server), which could facilitate concurrent, efficient predictions with minimal delay. Such an architecture would support sustained, high-volume encrypted traffic monitoring while maintaining performance and responsiveness.

Operational integration might require the development of real-time alerting mechanisms. By adopting standardized alert formats and establishing application programming interfaces or

connectors to Security Information and Event Management systems, Security Operations Center dashboards, or firewall systems, the system could provide actionable notifications within existing security workflows. Configurable alert thresholds can help balance detection sensitivity against false positives, thereby reducing alert fatigue. Incorporating Security Orchestration, Automation, and Response platforms could enable automated responses, such as blocking suspicious Internet Protocol addresses or terminating risky sessions, enhancing proactive threat mitigation. A user-friendly management interface for monitoring alerts, system status, and detection metrics may further improve usability and operational adoption.

Improving the interpretability of detection results remains a critical area for enhancement. Integrating explainable artificial intelligence techniques, such as SHapley Additive exPlanations or Local Interpretable Model-agnostic Explanations, could provide transparency by highlighting the features driving anomaly classifications, thus aiding analysts in understanding and verifying alerts. Expanding the system's capabilities to classify anomaly types through multi-task or hierarchical models may provide richer contextual information, facilitating faster incident response and forensic investigations. This shift from binary detection to detailed categorization could significantly increase the system's practical utility in security operations.

Finally, robust deployment may require iterative performance evaluation using realistic traffic datasets, including both benign and malicious encrypted flows, to validate throughput and detection accuracy. Continuous model retraining with up-to-date threat intelligence can ensure adaptability to evolving attack techniques. Comprehensive monitoring of system metrics such as latency, throughput, and false positive rates might support ongoing optimization and reliability improvements. Collectively, these strategies can transform the current prototype into a scalable, efficient, and explainable real-time anomaly detection system capable of meeting the stringent demands of modern cybersecurity environments.

5.9 Conclusion

This chapter summarized the development and evaluation of a deep learning-based system for detecting anomalies in encrypted HTTPS traffic using metadata alone. Although not deployed in a live setting, the system was effectively tested on structured, labelled datasets. The packet-based and session-based models achieved high test accuracies of 98.64% and 99.66%, and their combination resulted in an overall detection accuracy of 99.995%. These results confirm that

accurate, privacy-preserving anomaly detection is feasible without decrypting traffic. Despite current limitations—such as lack of real-time deployment and the inability to identify the specific type of anomaly—the system provides a strong foundation for future advancements in encrypted traffic security.

Chapter 6 : Conclusion and Recommendations

6.1 Conclusion

This research set out to address a critical gap in modern cybersecurity: the detection of anomalies in encrypted web traffic, specifically HTTPS, using deep learning methods that respect data privacy. The core objective was to develop a detection system that leverages traffic metadata—rather than decrypted content—to accurately identify abnormal behaviour. The study was driven by the hypothesis that a supervised deep learning model can detect anomalies in encrypted web traffic using only observable features such as flow duration, packet count, and TLS attributes. The system was designed to be scalable, modular, and privacy-preserving, aligning with the research goal of offering a practical solution suitable for real-world deployment. Traditional security tools are increasingly ineffective against encrypted threats due to their dependence on payload inspection. This study proposed, developed, and tested a deep learning-based anomaly detection system that analyses behavioural and metadata features instead of content, offering a privacy-preserving approach suitable for today's encrypted environments.

Two supervised classification models were developed: one trained on packet-based features such as packet count, byte count, and inter-arrival time, and the other on session-based features including flow duration, total packet/byte volume, and TLS-related attributes like Server Name Indication (SNI). These models were independently evaluated using a structured HTTPS dataset composed of both benign and malicious encrypted traffic. They demonstrated impressive performance, achieving test accuracies of 98.64% and 99.66%, respectively. By combining their outputs using a voting-based decision strategy, the overall detection accuracy of the system reached an exceptional 99.995%. This confirms the system's capability to identify a wide range of encrypted threats with minimal false positives.

The research also confirmed that deep learning models can generalize well from traffic metadata, learning to distinguish normal behaviour from threats without relying on payload decryption. This marks a significant step toward scalable, privacy-aware cybersecurity solutions that align with current industry needs. Although a live real-time deployment was not achieved due to hardware constraints, lack of access to specialized infrastructure for high-speed packet processing, and the complexity of deploying traffic capture tools in a production network, the prototype developed and tested in this dissertation provides a strong foundation for future full-scale implementation.

6.2 Recommendations

Based on the results of this research and the insights gained from the implementation process, the following recommendations are proposed for practitioners, researchers, and future developers of encrypted traffic anomaly detection systems:

i. **Implement in Privacy-Sensitive Domains:** Organizations operating in sectors where data confidentiality is paramount—such as healthcare, finance, and government—should adopt metadata-driven anomaly detection systems. These systems provide robust security capabilities without the need to decrypt user traffic, making them suitable for compliance with privacy regulations like GDPR and HIPAA.

ii. **Integrate with Existing Security Infrastructure:** The detection system should be deployed as part of a broader security architecture by integrating with Security Information and Event Management (SIEM) tools. This allows for centralized logging, correlation with other threat signals, automated alerting, and enhanced incident response.

iii. **Prioritize Dataset Expansion and Label Diversity:** Continued enrichment of the training dataset is essential to maintain model generalization. It is recommended that both benign and malicious traffic be updated regularly to reflect emerging threats, evolving encryption protocols, and changes in user behaviour patterns.

iv. **Enable Human-in-the-Loop Systems:** Incorporating feedback mechanisms from security analysts will allow models to evolve in real time. False positives can be corrected through expert review, and validated anomalies can be labelled to support incremental retraining and adaptive learning.

v. **Develop Anomaly Characterization Mechanisms:** The current system can flag anomalous behaviour but cannot identify its type or root cause. Post-detection modules should be introduced to classify anomalies into threat categories (e.g., ransomware, botnet, scanning) to improve response time and enhance threat intelligence.

vi. **Support Modular Deployment for Scalability:** Organizations adopting similar detection systems should use a modular architecture that allows components—such as data capture, preprocessing, classification, and alerting—to be deployed and scaled independently based on network size and operational requirements.

vii. **Conduct Cross-Environment Testing:** Before production deployment, it is recommended to test the models in various environments (e.g., enterprise networks,

cloud infrastructures, high-traffic public services) to ensure stability and performance consistency under diverse network conditions.

6.3 Future Work

Building on the current implementation and evaluation, several important directions have been identified to guide future development of the system toward full-scale, real-time deployment.

The most immediate priority is the integration of real-time traffic capture and detection capabilities. While the current prototype processes encrypted traffic in offline batches, the envisioned system should operate on live HTTPS streams. Achieving this requires the deployment of SPAN ports or passive network taps, real-time packet-to-flow conversion, and integration with low-latency inference pipelines. These components may benefit from hardware acceleration or parallel processing to ensure speed and scalability.

A second area of focus is the implementation of a dynamic decision fusion module that combines predictions from both packet-based and session-based models. While the current system uses a simple voting rule, future iterations should support flexible strategies, such as weighted scoring, confidence thresholds, and adaptive rule sets. Incorporating feedback loops from human analysts or response systems would also enable continuous learning and tuning of the detection logic.

Automating the alerting and response pipeline is another essential step. Integrating the system with SIEM platforms, firewalls, or intrusion prevention systems would allow the anomaly detection engine to transition from a passive observer to an active defence mechanism, capable of initiating alerts and enforcing policies in real time.

Future work should also address the interpretability gap. As the models currently detect the presence of an anomaly without specifying its type, a post-processing module capable of classifying anomalies by category or behaviour would significantly enhance the system's value for threat intelligence and forensic workflows. This may involve linking flagged instances to known threat signatures or clustering anomalies based on feature similarities.

Finally, future research should explore the use of advanced machine learning architectures, including ensemble models, attention mechanisms, and graph-based learning, to improve accuracy and generalization. Expanding the dataset to include more diverse benign and

malicious scenarios, and validating the system under varying network loads, will also be crucial to ensure reliability and operational robustness.

References

Agarwal, P., Swami, S., & Malhotra, S. (2022). Artificial Intelligence Adoption in the Post COVID-19 New-Normal and Role of Smart Technologies in Transforming Business: A Review. *Journal of Science and Technology Policy Management*, *ahead-of-print*. https://doi.org/10.1108/JSTPM-08-2021-0122

Althouse, J. B., Atkinson, J., & Atkins, J. (2017, June 25). *Open Sourcing JA3*. Salesforce Engineering. https://engineering.salesforce.com/open-sourcing-ja3-92c9e53c3c41/

Anderson, B., & McGrew, D. (2019). TLS Beyond the Browser: Combining End Host and Network Data to Understand Application Behaviour. *Proceedings of the Internet Measurement Conference*, 379–392. https://doi.org/10.1145/3355369.3355601

Bahlali, A. R., Bachir, A., & Cheriet, A. (2023). Malicious Encrypted Network Traffic Detection Using Deep Auto-Encoder with a Custom Reconstruction Loss. *2023 International Symposium on Networks, Computers and Communications (ISNCC)*, 1–7. https://doi.org/10.1109/ISNCC58260.2023.10323710

Behdadnia, T., Deconinck, G., Ozkan, C., & Singelee, D. (2023). Encrypted Traffic Classification for Early-Stage Anomaly Detection in Power Grid Communication Network. *2023 IEEE PES Innovative Smart Grid Technologies Europe (ISGT EUROPE)*, 1–6. https://doi.org/10.1109/ISGTEUROPE56780.2023.10407155

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, *13*(null), 281–305.

Bindu, P. V., & Thilagam, P. S. (2016). Mining social networks for anomalies: Methods and challenges. *Journal of Network and Computer Applications*, *68*, 213–229. https://doi.org/10.1016/j.jnca.2016.02.021

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly Detection: A Survey. *ACM Comput. Surv.*, *41*. https://doi.org/10.1145/1541880.1541882

Cheng, K., Wu, Z., Li, L., Wang, Q., Liu, T., Qiu, R., Liang, L., Ren, M., & Qin, C. (2023). Malicious Encryption Traffic Detection Based on Improved Convolutional Neural Network. *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, 1–5. https://doi.org/10.1109/ICMNWC60182.2023.10436019

Command and Control (C&C) Attacks. (n.d.). CrowdStrike.Com. Retrieved 27 March 2025, from https://www.crowdstrike.com/en-us/cybersecurity-101/cyberattacks/command-and-control-cac-attack/

Encrypted Web Traffic Dataset: Event Logs and Packet Traces. (n.d.). https://doi.org/10.57760/sciencedb.01676

Feghhi, S., & Leith, D. J. (2016). A Web Traffic Analysis Attack Using Only Timing Information. *IEEE Transactions on Information Forensics and Security*, *11*(8), 1747–1759. IEEE Transactions on Information Forensics and Security. https://doi.org/10.1109/TIFS.2016.2551203

Frolov, S., & Wustrow, E. (2019). The use of TLS in Censorship Circumvention. *Proceedings 2019 Network and Distributed System Security Symposium*. Network and Distributed System Security Symposium, San Diego, CA. https://doi.org/10.14722/ndss.2019.23511

García-Teodoro, P., Díaz-Verdejo, J., Maciá-Fernández, G., & Vázquez, E. (2009). Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, *28*(1), 18–28. https://doi.org/10.1016/j.cose.2008.08.003

Goel, S., Guleria, K., & Panda, S. N. (2022). Anomaly based Intrusion Detection Model using Supervised Machine Learning Techniques. *2022 10th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 1–5. https://doi.org/10.1109/ICRITO56286.2022.9965050

Grammatikis, P. R., & Sarigiannidis, P. (2021). Network Threats. In *Cyber-Security Threats, Actors, and Dynamic Mitigation*. CRC Press.

Han, S., Wu, Q., Zhang, H., & Qin, B. (2022). Light-weight Unsupervised Anomaly Detection for Encrypted Malware Traffic. *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*, 206–213. https://doi.org/10.1109/DSC55868.2022.00034

He, G., Yang, M., Gu, X., Luo, J., & Ma, Y. (2014). A novel active website fingerprinting attack against Tor anonymous system. *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, 112–117. https://doi.org/10.1109/CSCWD.2014.6846826

He, L., & Shi, Y. (2018). *Identification of SSH Applications Based on Convolutional Neural Network*. 198–201. https://doi.org/10.1145/3230348.3230458

Hevner, March, Park, & Ram. (2004). Design Science in Information Systems Research. *MIS Quarterly*, *28*(1), 75. https://doi.org/10.2307/25148625

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the Dimensionality of Data with Neural Networks. *Science*, *313*(5786), 504–507. https://doi.org/10.1126/science.1127647

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. Neural Computation. https://doi.org/10.1162/neco.1997.9.8.1735

Ibraheem, H. R., Zaki, N. D., & Al-mashhadani, M. I. (2022). Anomaly detection in encrypted HTTPS traffic using machine learning: A comparative analysis of feature selection techniques. *Mesopotamian Journal of Computer Science*, *2022*, 18–28. https://doi.org/10.58496/MJCSC/2022/005

Isingizwe, D. F., Wang, M., Liu, W., Wang, D., Wu, T., & Li, J. (2021). Analyzing Learning-based Encrypted Malware Traffic Classification with AutoML. *2021 IEEE 21st International Conference on Communication Technology (ICCT)*, 313–322. https://doi.org/10.1109/ICCT52962.2021.9658106

Karagiannis, T., Papagiannaki, K., & Faloutsos, M. (2005). BLINC: Multilevel traffic classification in the dark. *ACM SIGCOMM*, *35*(4), 229–240. https://escholarship.org/uc/item/1wn9n8kt

Lashkari, A. H., Gil, G. D., Mamun, M. S. I., & Ghorbani, A. A. (2024). *Characterization of Tor Traffic using Time based Features*. 253–262. https://www.scitepress.org/Link.aspx?doi=10.5220/0006105602530262

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Long, G., & Zhang, Z. (2023). Deep Encrypted Traffic Detection: An Anomaly Detection Framework for Encryption Traffic Based on Parallel Automatic Feature Extraction. *Computational Intelligence and Neuroscience*, *2023*(1), 3316642. https://doi.org/10.1155/2023/3316642

Luo, W., Liu, Z., Zhao, R., Chen, J., & Deng, X. (2021). Malicious HTTPS Traffic Classification Algorithm Based on DCGAN_1D-CNN. *2021 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS)*, 20–25. https://doi.org/10.1109/TOCS53301.2021.9688753

Ma, H., & Zhang, Z. (2020). A New Private Information Encryption Method in Internet of Things under Cloud Computing Environment. *Wireless Communications and Mobile Computing*, *2020*(1), 8810987. https://doi.org/10.1155/2020/8810987

Moore, A. W., & Zuev, D. (2005). Internet traffic classification using bayesian analysis techniques. *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 50–60. https://doi.org/10.1145/1064212.1064220

Pandey, A. B., Tripathi, A., & Vashist, P. C. (2022). A Survey of Cyber Security Trends, Emerging Technologies and Threats. In R. Agrawal, J. He, E. Shubhakar Pilli, & S. Kumar (Eds.), *Cyber Security in Intelligent Computing and Communications* (pp. 19–33). Springer. https://doi.org/10.1007/978-981-16-8012-0_2

Pang, G., Shen, C., Cao, L., & Hengel, A. van den. (2022). Deep Learning for Anomaly Detection: A Review. *ACM Computing Surveys*, *54*(2), 1–38. https://doi.org/10.1145/3439950

Patcha, A., & Park, J.-M. (2007). An overview of anomaly detection techniques: Existing solutions and latest technological trends. *Computer Networks*, *51*(12), 3448–3470. https://doi.org/10.1016/j.comnet.2007.02.001

Pfleeger, C. P., & Pfleeger, S. L. (2012). *Analyzing Computer Security: A Threat/vulnerability/countermeasure Approach*. Prentice Hall Professional.

Rescorla, E. (2018). *The Transport Layer Security (TLS) Protocol Version 1.3* (Request for Comments RFC 8446). Internet Engineering Task Force. https://doi.org/10.17487/RFC8446

Rescorla, E., & Dierks, T. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2* (Request for Comments RFC 5246). Internet Engineering Task Force. https://doi.org/10.17487/RFC5246

Roesch, M. (1999). Snort—Lightweight Intrusion Detection for Networks. *Proceedings of the 13th USENIX Conference on System Administration*, 229–238.

Rosa, L., Cruz, T., Borges de Freitas, M., Quitério, P., Henriques, J., Caldeira, F., Monteiro, E., & Simoes, P. (2021). Intrusion and anomaly detection for the next-generation of industrial automation and control systems. *Future Generation Computer Systems*. https://doi.org/10.1016/j.future.2021.01.033

Sadqi, Y., & Mekkaoui, M. (2021). Design Challenges and Assessment of Modern Web Applications Intrusion Detection and Prevention Systems (IDPS). In M. Ben Ahmed, İ. Rakıp Karaș, D. Santos, O. Sergeyeva, & A. A. Boudhir (Eds.), *Innovations in Smart Cities Applications Volume 4* (pp. 1087–1104). Springer International Publishing. https://doi.org/10.1007/978-3-030-66840-2_83

Sah, A. K., & K, V. (2024). Anomaly-Based Intrusion Detection in Network Traffic using Machine Learning: A Comparative Study of Decision Trees and Random Forests. *2024 2nd International Conference on Networking and Communications (ICNWC)*, 1–7. https://doi.org/10.1109/ICNWC60771.2024.10537451

Schmidl, S., Wenig, P., & Papenbrock, T. (2022). Anomaly detection in time series: A comprehensive evaluation. *Proceedings of the VLDB Endowment*, *15*(9), 1779–1797. https://doi.org/10.14778/3538598.3538602

Seth, B., Dalal, S., Jaglan, V., Le, D.-N., Mohan, S., & Srivastava, G. (2022). Integrating encryption techniques for secure data storage in the cloud. *Transactions on Emerging Telecommunications Technologies*, *33*(4), e4108. https://doi.org/10.1002/ett.4108

Shaukat Dar, K., Mahboob Alam, T., Luo, S., Shabbir, S., Hameed, I., Li, J., Abbas, S., & Javed, U. (2021). *A Review of Time-Series Anomaly Detection Techniques: A Step to Future Perspectives*. https://doi.org/10.1007/978-3-030-73100-7_60

Singh, A. P., & Singh, M. (2021). A Comparative Review of Malware Analysis and Detection n HTTPs Traffic. *International Journal of Computing and Digital Systems*, *10*(1), 111–123. https://doi.org/10.12785/ijcds/100111

Špaček, S., Velan, P., Čeleda, P., & Tovarňák, D. (2022). Encrypted Web traffic dataset: Event logs and packet traces. *Data in Brief*, *42*, 108188. https://doi.org/10.1016/j.dib.2022.108188

Thapa, S., & Mailewa, A. (2020, April 3). *THE ROLE OF INTRUSION DETECTION/PREVENTION SYSTEMS IN MODERN COMPUTER NETWORKS: A REVIEW*.

Tulbure, A.-A., Tulbure, A.-A., & Dulf, E.-H. (2022). A review on modern defect detection models using DCNNs – Deep convolutional neural networks. *Journal of Advanced Research*, *35*, 33–48. https://doi.org/10.1016/j.jare.2021.03.015

Ucci, D., Sobrero, F., Bisio, F., & Zorzino, M. (2021). Near-real-time Anomaly Detection in Encrypted Traffic using Machine Learning Techniques. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 01–08. https://doi.org/10.1109/SSCI50451.2021.9659955

Vikram, A., & Mohana. (2020). *Anomaly detection in Network Traffic Using Unsupervised Machine learning Approach*. 476–479. https://doi.org/10.1109/ICCES48766.2020.9137987

Wang, B., Guan, Y., Gou, G., Fu, P., Li, Z., Yang, Q., & Liu, C. (2022). MTBD: HTTPS Tunnel Detection Based on Multi-dimension Traffic Behaviours Decision. *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application*

(HPCC/DSS/SmartCity/DependSys), 474–481. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys57074.2022.00091

Wang, L., Dyer, K. P., Akella, A., Ristenpart, T., & Shrimpton, T. (2015). Seeing through Network-Protocol Obfuscation. *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 57–69. https://doi.org/10.1145/2810103.2813715

Wang, W., Zhu, M., Wang, J., Zeng, X., & Yang, Z. (2017). End-to-end encrypted traffic classification with one-dimensional convolution neural networks. *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, 43–48. https://doi.org/10.1109/ISI.2017.8004872

Wang, Z., & Thing, V. (2022). *Encrypted Traffic Feature Dataset for Machine Learning and Deep Learning based Encrypted Traffic Analysis*. *1*. https://doi.org/10.17632/xw7r4tt54g.1

Xing, J., & Wu, C. (2020). Detecting Anomalies in Encrypted Traffic via Deep Dictionary Learning. *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 734–739. https://doi.org/10.1109/INFOCOMWKSHPS50562.2020.9162940

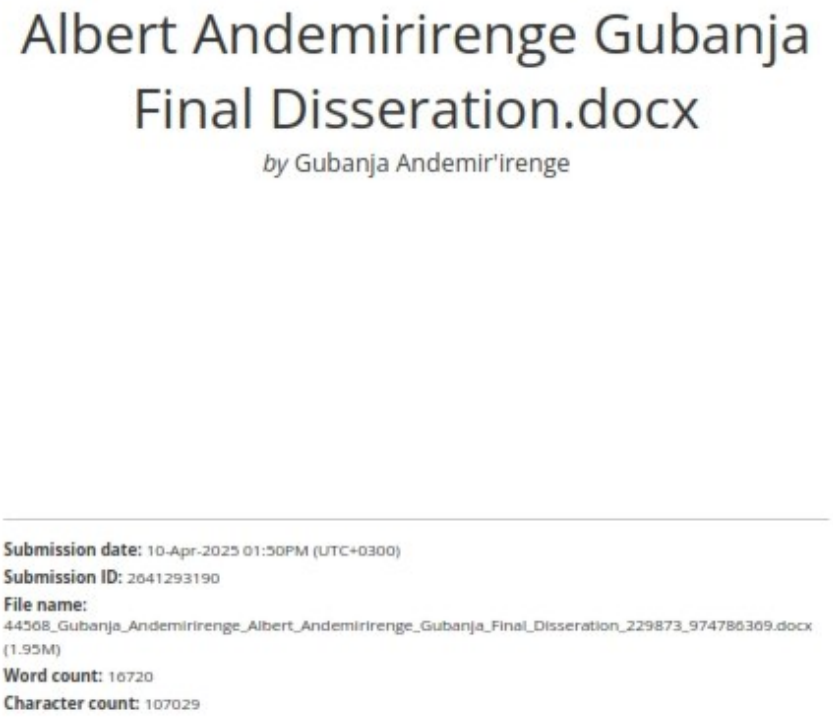
Yang, J., & Lim, H. (2021). Deep Learning Approach for Detecting Malicious Activities Over Encrypted Secure Channels. *IEEE Access*, *9*, 39229–39244. IEEE Access. https://doi.org/10.1109/ACCESS.2021.3064561

Yu, T., Zou, F., Li, L., & Yi, P. (2019). An Encrypted Malicious Traffic Detection System Based on Neural Network. *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 62–70. https://doi.org/10.1109/CyberC.2019.00020

Zaid, T., & Garai, S. (2024). Emerging Trends in Cybersecurity: A Holistic View on Current Threats, Assessing Solutions, and Pioneering New Frontiers. *Blockchain in Healthcare Today*, *7*, 10.30953/bhty.v7.302. https://doi.org/10.30953/bhty.v7.302

Appendices

Appendix A: Similarity Report



Albert Andemirirenge Gubanja Final Disseration.docx

ORIGINALITY REPORT

13%	11%	13%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	arxiv.org Internet Source	1%
2	ouci.dntb.gov.ua Internet Source	1%
3	Bingxu Wang, Yangyang Guan, Gaopeng Gou, Peipei Fu, Zhen Li, Qingya Yang, Chang Liu. "MTBD: HTTPS Tunnel Detection Based on Multi-dimension Traffic Behaviors Decision", 2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys), 2022 Publication	1%
4	su-plus.strathmore.edu Internet Source	1%
5	dergipark.org.tr Internet Source	<1%
6	link.springer.com Internet Source	<1%
7	Submitted to University of Wales Swansea Student Paper	<1%

Appendix B: Ethical Clearance Confirmation

Strathmore UNIVERSITY

28th March 2025

Mr Gubanja Albert,
gubanja.andemir'irenge@strathmore.edu

Dear Mr Gubanja,

RE: Anomaly Detection in Encrypted Web Traffic Using Deep Learning Mechanisms

This is to inform you that SU-ISERC has reviewed and **approved** your above **SU-masters** proposal. Your application reference number is **SU-ISERC2794/25.** The approval period is from **28th March 2025 to 27th March 2026.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used.
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-ISERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-ISERC within 72 hours of notification.
iv. Any changes anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-ISERC within 72 hours.
v. Clearance for the export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to the expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days of completion of the study to SU-ISERC.

Before commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology, and Innovation (NACOSTI) https://research-portal.nacosti.go.ke/ and obtain other clearances needed.

Yours sincerely,

Mr Ambrose Rachier,
Chairperson; SU-ISERC

Ole Sangale Rd, Madaraka Estate. PO Box 59857-00200, Nairobi, Kenya. Tel +254 (0)703 034000
Email admissions@strathmore.edu www.strathmore.edu