

DEEP LEARNING APPROACHES TO PROFILING ORGANIZATIONAL THREATS — NEXTGEN SOC

Project ID: 24-25J-075
Group Project Report

BSc. (Hons) in Information Technology Specializing in Cybersecurity

Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

April 2025

DEEP LEARNING APPROACHES TO PROFILING ORGANIZATIONAL THREATS – NEXTGEN SOC

Project ID: 24-25J-075

Supervisor: Dr. Harinda Fernando

Co Supervisor: Mr. Kavinga Yapa Abeywardena

BSc. (Hons) in Information Technology Specializing in Cybersecurity

Department of Computer Systems Engineering




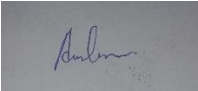
Sri Lanka Institute of Information Technology

Sri Lanka

April 2025

DECLARATION

I declare that this is my own work, and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or Diploma in any other University or institute of higher learning, and to the best of my knowledge and belief, it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

NAME	STUDENT ID	SIGNATURE
D.H. Senevirathna	IT21298608	
W.M.M. Gunasekara	IT21226496	
M.F.F. Ashra	IT21380396	
K.P.A.T. Gunawardhana	IT21058196	

Name of supervisor: Dr. Harinda Fernando

Name of co-supervisor: Mr. Kavinga Yapa Abeywardena

The above candidates have carried out research for the bachelor's degree dissertation under my supervision.

Signature:

Date: 2025.04.

Supervisor: Dr. Harinda Fernando

Signature:

Date: 2025.04.

Co Supervisor: Mr. Kavinga Yapa Abeywardena

ABSTRACT

Organizations face more complex and dynamic threats in the quickly changing cybersecurity landscape than can be handled by conventional Security Operations Centers (SOCs). In response, this study suggests a deep learning-powered the next-generation SOC (NextGen SOC) platform to improve organizational threat profiling. By combining four essential data dimensions, network traffic, endpoint activity, human behavior, and physical security, the main goal is to create a strong, flexible system that can identify, evaluate, and forecast dangers in real time.

The suggested solution is flexible and integrates different deep learning models into each of its parts. Using tools like Npcap, Scapy, and Socket.IO, the Network Traffic Analysis module classifies real packet flows using Generative Adversarial Networks (GANs) and Feedforward Neural Networks (FNNs). This component, which was used for training on the CICIDS 2017 dataset, identified abnormal traffic with an accuracy rate of 96.39%. High performance was shown on the KDD dataset by the Endpoint Threat Detection module, which processes endpoint activity and system logs using Random Forest as well as Neural Network models. Real-time outputs are integrated into a Flask-based dashboard.

The system uses Convolutional Neural Networks (CNNs) including ensemble learning techniques to profile user activity for Human Behavior Analysis, with an emphasis on psychological features, device access, and login behavior. It achieved a 95% detection rate with the CERT Insider Threat Dataset. The Physical Security Threat Detection module, which was trained on the Actual Violence Situations Dataset and achieved 99.16% accuracy on lightweight infrastructure, uses Transfer Learning using MobileNetV2 and LSTM layers to examine CCTV footage for violent behavior.

Real-time dashboards are available for all modules to improve situational awareness and usability. This study shows that a unified, based on artificial intelligence SOC framework that offers high scalability, reliability, and adaptability for actual organizational security is both feasible and practical.

Keywords: next-generation SOC (NextGen SOC), Insider Threat Detection, Real-Time Monitoring, Endpoint Threat Detection, Network Analysis

ACKNOWLEDGMENT

We want to thank everyone who helped and advised us during the successful conclusion of this research project, which is named *"Deep Learning Approaches to Profiling Organizational Threats – NEXTGEN SOC."*

Firstly, we are deeply appreciative of our supervisor and co-supervisor, whose perceptive feedback, unwavering support, and professional expertise were invaluable in moulding this study. Under their direction, we were able to overcome several challenges and enhance the standard and scope of our work.

We also want to express our gratitude to the mentors and academic staff at the Sri Lanka Institute of Information Technology (SLIIT) for giving us the tools, resources, and expertise we needed to complete this project.

We would like to express our sincere gratitude to our peers and colleagues, whose diligence, cooperation, and exchange of ideas aided in the overall creation of this system.

Finally, for their unfailing love, tolerance, and moral support along this journey, we are particularly appreciative of our parents and relatives. Their comprehension and encouragement enabled us to stay resolute and focused.

The achievement might not have been achieved without all of your help.

TABLE OF CONTENTS

DECLARATION	iii
ABSTRACT	iv
ACKNOWLEDGMENT	v
TABLE OF CONTENTS	vi
LIST OF FIGURES	ix
LIST OF TABLES	x
LIST OF ABBREVIATIONS	xi
1. INTRODUCTION	1
1.1. Research Background	1
1.2. Research Problem Statement	2
1.3. Significant of the Study	3
1.4. Research Aim.....	4
1.5. Research Objectives	4
1.5.1. Main Objective.....	4
1.6. Research Scope	5
2. LITERATURE REVIEW	8
2.1. Network Traffic Analysis.....	8
2.2. Human Behavior Analysis	10
2.3. Endpoint Security Analysis	13
2.4. Physical Security Analysis.....	17
3. METHODOLOGY	20
3.1. Introduction to Methodology.....	20
3.2. Overall System Design and Architecture	21
3.3. Threat Profiling with Network Traffic Analysis	22
3.3.1. Dataset Description	23
3.3.2. Methodological Structure.....	23
3.3.3. Research Architecture	24
3.3.4. Component Diagram	27
3.3.5. Functional Requirements	28

3.3.6. Non-Functional Requirements.....	29
3.3.7. Used Tools and Technologies	29
3.4. Threat Profiling with Human Behavior Analysis.....	29
3.4.1. Dataset Description	29
3.4.2. Methodological Structure.....	30
3.4.3. Research Architecture	35
3.4.4. Component Diagram	37
3.4.5. Functional Requirements	38
3.4.6. Non – Functional Requirements	39
3.4.7. Used Tools and Technologies	39
3.5. Threat Profiling with Endpoint Security Analysis	41
3.5.1. Dataset Description.....	41
3.5.2. Methodological Structure.....	41
3.5.3. Research Architecture	43
3.5.4. Component Diagram	44
3.5.5. Functional Requirements	45
3.5.6. Non-Functional Requirements.....	45
3.5.7. Used Tools and Technologies	46
3.6. Threat Profiling with Physical Security Analysis	46
3.6.1. Dataset Description	46
3.6.2. Methodological Structure	47
3.6.4. Component Diagram	53
3.6.5. Functional Requirements.....	54
3.6.6. Non-Functional Requirements	55
3.6.7. Used Tools and Technologies	56
3.7. Software Architecture Model	57
4. IMPLEMENTATION AND TESTING	60
4.1. Implementation.....	60
4.2. Threat Profiling with Network Traffic Analysis	60
4.2.1. Data Preprocessing and Feature Engineering	60
4.2.2. Model Development.....	60

4.2.3. System Testing and Validation	62
4.3. Threat Profiling with Human Behavior Analysis	63
4.3.1. Data Preprocessing and Feature Engineering	63
4.3.2. Model Development.....	63
4.3.3. System Testing and Validation	64
4.4. Threat Profiling with Endpoint Security Analysis	65
4.4.1. Data Preprocessing and Feature Engineering	65
4.4.2. Model Design Development.....	65
4.4.3. System Testing and Validation	66
4.5. Threat Profiling with Physical Security Analysis	67
4.5.1. Data Preprocessing and Feature Engineering	67
4.5.2. Model Development.....	67
5. RESULT AND DISCUSSION	69
5.1. Network Traffic Analysis.....	69
5.1.1. Research Findings	69
5.1.2. Discussion.....	72
5.2. Human Behavior Analysis	73
5.2.1. Research Findings	73
5.2.2. Discussion.....	76
5.3. Endpoint Security Analysis	78
5.3.1. Research Findings.....	78
5.3.2. Discussion	80
5.4. Physical Security Analysis	81
6. COMMERCIALIZATION.....	84
7. BUDGET ALLOCATION	87
8. GANTT CHART.....	88
9. CONTRIBUTION.....	89
10. CONCLUSION	90
11. REFERENCES	92
12. APPENDICES	96

LIST OF FIGURES

Figure 1 - Overall System Diagram	21
Figure 2 - Dataset (Network Traffic Analysis)	23
Figure 3 - Network Traffic Analysis Component Diagram	27
Figure 4 - Network Traffic Analysis Flow Diagram	28
Figure 5 – Dataset (Human Behavior Analysis)	30
<i>Figure 6 - Insider Threat Detection Component Diagram</i>	37
Figure 7 - Dataset (Endpoint Security Analysis)	41
Figure 8 - Component Diagram (Endpoint Threat Detection System)	44
Figure 9- Component Diagram (Physical Security System)	53
Figure 10- SDLC Methodology Life Cycle.	59
Figure 11 - Accuracy and Loss over Epochs (Network Traffic)	70
Figure 12 - Confusion Matrix(Network Traffic)	70
Figure 13 - Model Accuracy Comparison	75
Figure 14 - Accuracy of the CNN Model	75
Figure 15 - Loss of Accuracy	75
Figure 16 - Comparison of Model Performance with Existing Literature (Human Behavior)	76
Figure 17 - Model Accuracy Comparison	79
Figure 18 - Gantt Chart	88

LIST OF TABLES

Table 1- List of Abbreviation.....	xi
Table 2 - Research Gap of Network Traffic Analysis	10
Table 3- Research Gap of Human Behavior Analysis	13
Table 4 - Research Gap (Endpoint Security).....	16
Table 5- Research Gap (Physical Security Analysis)	19
Table 6 –Classification Report (Network Traffic)	71
Table 7 - Comparison of Model Performance with Existing Literature (Network Traffic)....	72
Table 8 - CNN model Evaluation Results	74
Table 9 - Budget Allocation.....	87
Table 10 - Contribution of each Student	89

LIST OF ABBREVIATIONS

Abbreviation	Full Form
SOC	Security Operations Centers
ML	Machine learning
DL	Deep learning
GAN	Generative Adversarial Networks
FNN	Feedforward Neural Networks
CNN	Convolutional Neural Networks
AI	Artificial Intelligence
LSTM	Long Short-Term Memory
XAI	Explainable Artificial Intelligence
IDS	Intrusion Detection System
DDoS	Distributed Denial-of-Service
API	Application Programming Interface
IoT	Internet of Things
IDE	Integrated Development Environment
LKR	Sri Lankan Rupee

Table 1- List of Abbreviation

1. INTRODUCTION

1.1. Research Background

Over the past ten years, the cybersecurity environment has rapidly changed, and organizations now have to contend with an unprecedented rise in the scope and complexity of cyberthreats. Once successful in handling traditional attacks using rule-based as well as signature-based detection systems, traditional Security Operations Centers (SOCs) are becoming less and less capable of handling contemporary dangers like advanced persistent threats (APTs), insider threats, and zero-day exploits [1].

The transition from reactive surveillance to proactive and intelligent threat management is reflected in the evolution of SOCs. Predefined rules and manual analysis were the mainstays of legacy SOCs, which resulted in poor reaction times and high false positive rates. Next-Generation SOCs (NextGen SOCs) are intelligent, scalable systems that make use of robotics, machine learning (ML), along with deep learning (DL) technologies as a result of these constraints [2]. But creating such systems is not without its challenges. One of the biggest ones is the volume and diversity of data: modern organizations generate a lot of different types of data from different sources, such as network traffic, endpoint logs, monitoring systems, and employee behavior, and it is still very difficult to integrate and analyze this data in real time.

The changing threat scenario presents another difficulty. Cybercriminals are always evolving, producing new software and attack techniques to circumvent established defenses. SOCs must therefore use adaptive learning models that change in response to new threats. Real-time processing and scalability are also essential; as organizational infrastructure grows, so too, the SOC's capacity to track anomalies and react quickly without experiencing performance problems [3].

The combination of monitoring and behavioral data also raises ethical and privacy issues. Maintaining user confidentiality and ensuring ethical management of confidential data is crucial, particularly as firms implement additional monitoring technologies for increased security [4].

In order to solve these problems, this study suggests a deep learning-based methodology for identifying organizational hazards. The project creates a single threat detection architecture by

combining several data sources, including network traffic, endpoint logs, physical security monitoring, and behavioral data from people. Enhancing threat prediction accuracy, decreasing false positives, and improving overall responsiveness, the suggested system makes use of Generative Adversarial Networks (GANs), Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), Random Forests, and transfer learning. The project's final objective is to aid in the creation of a NextGen SOC that is ethical, intelligent, and scalable and that can use automatic reaction mechanisms and real-time analytics to protect against modern cyberthreats.

1.2. Research Problem Statement

The complexity and interconnectedness of digital infrastructures expose organizations to a greater variety of cyberthreats. Conventional Security Operations Centers (SOCs) frequently rely on rule-based detection systems and isolated data sources, and they frequently function in isolation [5]. This disjointed strategy reduces the SOC's overall efficacy by making it challenging to correlate danger signals across multiple domains, including network traffic, endpoint activity, human behavior, and physical security systems.

Cyber risks are also becoming more common, complex, and ever-changing. Insider threats, zero-day attacks, and multi-vector incursions that circumvent traditional security procedures are examples of contemporary threats. Due to a number of constraints, the current SOC frameworks find it difficult to adapt to this changing environment [6]. These include poor response times, high false-positive rates, limited real-time adaptation to novel and unheard-of attack patterns, and the inability to combine disparate data sources. Furthermore, the majority of systems are ineffective at managing encrypted or fast network traffic and do not adequately analyze human behavioral indicators, which are indicators that frequently indicate insider threats.

Because of this, current SOC's have a tough time continuing to detect threats effectively and respond to incidents promptly, which increases security risks. Designing and implementing Next-Generation SOC's that make use of deep learning technology is crucial in order to solve these mounting concerns. These systems need to be able to combine threat intelligence from several input sources and analyze enormous amounts of varied data in real time.

The goal of this project is to develop a combined threat profiling system which detects organizational threats accurately, in real time, and with context awareness by leveraging innovative machine learning models. In order to eradicate fragmentation and improve the speed and accuracy of cyber threat response in contemporary enterprises, the suggested approach integrates data from network activity, endpoint devices, user behavior, and physical security systems into a unified analytical framework.

1.3. Significant of the Study

This study is important because it helps modern Security Operations Centers (SOCs) become more intelligent, flexible, and responsive. The complexity and volume of cyber threats are increasing, making it difficult for standard security measures to provide prompt and precise threat identification. This study proposes a deep learning-driven approach for comprehensive threat profiling across several domains, thereby filling a crucial gap in existing cybersecurity methods.

This paper presents a comprehensive method for detecting organizational threats by combining network traffic, endpoints logs, human behavioral patterns, as well as physical security records into a single framework. A more accurate and proactive way to identify both external and insider threats is to integrate advanced models like Generative Adversarial Networks (GANs), Feedforward Neural Networks (FNNs), Convolutional Neural Networks (CNNs), Transfer Learning, and ensemble learning techniques. In addition to increasing threat visibility, this interdisciplinary approach lowers false positives and improves real-time decision-making.

Additionally, the report encourages the adoption of real-time dashboards and ethical AI methods to help cybersecurity teams make well-informed decisions. Because of its modular nature and scalability, the system may be used in a variety of organizational settings and provides a creative and useful model for creating NextGen SOCs.

By highlighting the efficiency of deep learning techniques in organizational threat profiling, the research findings are anticipated to make a substantial contribution to the domains of artificial intelligence and cybersecurity. In the end, this study encourages the creation of more intelligent, flexible security systems that can keep up with the quickly evolving threat landscape, improving the protection of vital organizational data and assets.

1.4. Research Aim

The goal of this research is to combine data collected from network traffic, endpoint devices, human behavior, and physical security systems to create a deep learning-based threat profile system for Next-Generation Security Operations Centers (SOCs). In addition to lowering false positives and increasing the accuracy and sensitivity of SOCs in detecting internal and external threats, this unified architecture aims to enhance real-time threat detection.

1.5. Research Objectives

1.5.1. Main Objective

The primary goal of this research is to use innovative deep learning models to develop and deploy an integrated, real-time threat profiling system for Next-Generation Security Operations Centers (SOCs). In order to improve cybersecurity monitoring and response, the system analyses several data sources, including network traffic, endpoint activity, human behavior, and physical security systems, in order to precisely identify and forecast organizational threats.

1.5.2. Sub Objectives

- To create a system for detecting network threats in real time by classifying traffic using GANs and FNNs.
- Use CNN and Random Forest models to examine user behavior and find insider threats.
- To create an endpoint threat detection system based on deep learning for traffic and system log analysis.
- To incorporate LSTM for physical threat analysis and MobileNetV2 for video surveillance threat detection.
- To implement a single web-based dashboard that allows for real-time visualization and threat detection and response.
- To guarantee the system's accuracy, scalability, and adaptability in actual business settings.

1.6. Research Scope

In order to address the changing demands of contemporary organizational environments where threats are becoming more intricate, multifaceted, and sophisticated, this research focuses on developing a holistic cybersecurity solution. The technical scope of this project is its main component. It uses a variety of sophisticated deep learning methods, such as Generative Adversarial Networks (GANs), Convolutional Neural Networks (CNNs), conventional Feedforward Neural Networks (FNNs), and Random Forest classifiers, to identify, evaluate, and categorize various organizational threats in real time. A comprehensive approach to threat identification and response is ensured by the application of these techniques across several security monitoring levels, including user behavior, digital network traffic, and physical security surveillance.

The integration of several excellent datasets that represent various facets of organizational operations and security is included in the research's data scope. While the CERT Insider Threat dataset aids in modelling insider threats by examining user activity, personality attributes, and access behavior, the CICIDS2017 dataset is used for training models in detecting network-based anomalies along with cyber intrusions [7]. System applications and activity logs are used for endpoint threat analysis in order to spot questionable trends. Additionally, by analyzing video surveillance data in real-time, the Real-Life Violence Situations Dataset helps the system identify physical security hazards. The system's ability to comprehend and react to threats from many vectors is ensured by this diversity of data sources.

The research's functional scope is to develop a single, intelligent Security Operations Centre (SOC) interface that consolidates threat data from various networks, endpoints, behavioral, and physical components. The solution gives cybersecurity experts the ability to make quick, well-informed judgements by providing real-time visualizations, alerts, and summaries through a flexible, web-based dashboard. For instant situational awareness and response, each subsystem analyses incoming data, applies the relevant model, and relays the findings to this central dashboard.

Lastly, the deployment scope includes putting the system into practice in both real-world organizational contexts and controlled, lab-based situations. This makes it possible to thoroughly verify the system's accuracy, adaptability, scalability, and dependability. The objective is to make sure the solution is reliable enough to manage the changing dynamics of actual organizational infrastructures in addition to performing effectively under experimental

settings. The research will confirm the viability and efficacy of the suggested integrated SOC system in improving threat identification and incident handling capabilities through this phased deployment.

1.3. Target Audience

By filling existing gaps in detection, response, and evaluation with deep learning technologies, this project aims to help a broad spectrum of stakeholders in the security and technology sectors. The following important groups are part of the target audience:

- **SOC cybersecurity experts and analysts:** In charge of identifying and countering cyberthreats, these people are on the front lines of organizational defense. The suggested solution provides AI-powered threat detection capabilities that improve their capacity to track and respond to security events instantly. Through the integration of several threat sources, including network traffic, endpoint activity, human behavior, and physical security, the system offers SOC teams a more complete situational awareness platform.
- **System administrators and IT managers:** IT executives are responsible for guaranteeing the dependability, availability, and security of the IT infrastructure within their organizations. They receive a scalable, smart solution from this study that enhances visibility across several systems and aids in the proactive detection of anomalies or vulnerabilities. Predictive analytics and a real-time monitoring dashboard facilitate quicker decision-making and lower the possibility of security breaches.
- **Researchers and academics:** One important academic contribution is the incorporation of GANs, CNNs, Random Forests, and various other deep learning models into threat profiling. Because it provides thorough techniques, real-world implementations, and performance assessments that researchers can expand upon or compare to, this study is a useful resource for future research in AI-based cybersecurity.

- **Software developers and system integrators:** The modular design and technological stack of this research can be used by developers and integrators who are working on custom security platforms or improving current systems. The system's extensible and interoperable components, which were constructed with Flask, TensorFlow, Socket.IO, and Scapy, make it appropriate for incorporation into bigger cybersecurity ecosystem or proprietary security platforms.
- **Institutions of government and business:** To strengthen security posture, national agencies, military units, and business organizations with vital infrastructure can implement this system. Real-time threat detection from both the digital and physical realms helps ensure regulatory compliance, safeguards confidential information, and helps protect both digital assets as well as physical properties. Because of its robust and scalable design, the system can be implemented in high-security settings.

2. LITERATURE REVIEW

The increasing level of sophistication and complexity of cyber-attacks has led to a considerable evolution in Security Operations Centers (SOCs). With little automation as well as restricted real-time threat detection capabilities, the first generation of SOCs (Gen 1) mostly relied on manual monitoring of security system warnings and system logs [8]. The second generation (Gen 2) deployed security information and event management (SIEM) systems to centralize data gathering and correlation as cybersecurity demands grew. These systems still had problems like fatigue from alerts and the inability to adjust to new dangers, even if this made systematic analysis and alerting possible [9]. A meaningful change was brought about by the transition to Next-Generation SOCs (NG-SOCs), which combined automation, behavioral analytics, machine learning, and artificial intelligence. NG-SOCs place a strong emphasis on automated response procedures, cross-domain visibility, and proactive threat hunting. Additionally, they integrate extended detection and response (XDR) abilities, which offer a consolidated perspective of security events across people, networks, endpoints, and cloud environments [10].

2.1. Network Traffic Analysis

Generative Adversarial Networks (GANs) are becoming a major application in intrusion detection systems (IDS) for countering the problem of detecting rare or zero-day attacks. Yin et al. (2022) presented NetShare, a GAN-based method to generate IP header traces, which improved detection generalization [11]. Training instability and scalability problems, though reported by the study, make it less suitable for real-time applications. Thapa et al. (2020) compared various ML and DL models and stated that GAN-augmented datasets enhance model resilience to unseen attacks but at the expense of extensive tuning and resource allocation [12]. Hindy et al. (2020) used deep learning autoencoders to detect zero-day attacks using the CICIDS2017 dataset. While their approach proved efficacy in anomaly detection, it lacked consistency across different attack types and suffered limits in generalization, particularly when dealing with imbalanced datasets [13].

This work diverges from computationally costly GAN usage by applying a lightweight GAN as a purpose for offline data augmentation. The GAN utilized in this work entails two neural

networks: a generator, where it creates artificial network traffic samples using random noise, and a discriminator, where it is learned to identify whether traffic is synthetic or real. Using adversarial training, the generator improves its ability to mimic real attack data. After training, the generator produced synthetic samples, which were used to augment the CICIDS 2017 dataset. This helps in avoiding the issue of class imbalance and improving the generalization ability of the model, especially towards less represented and zero-day attacks. Unlike previous work that incorporates GANs into the detection pipeline, this paper limits GAN use to training preserving runtime efficiency while enhancing model robustness.

Feedforward Neural Networks (FNNs) are used in NIDS because they are of low computational expense, easy to use, and work very effectively with binary classification problems. Vinayakumar et al. (2020) proved that FNNs are effective against brute-force and port scan attacks, especially when it comes to limited resource [14]. Their model, however, lacked generality in cases of generalizing unknown attacks due to a lack of diversity in data. Ahmed et al. (2022) found that SMOTE enhanced the overall accuracy of Feedforward Neural Networks (FNNs) in intrusion detection but that the approach continued to struggle to detect rare or minority class types of attacks [15].

This paper builds on these foundations by leveraging actual training data combined with GAN-generated synthetic samples to allow the FNN to learn uncommon attack patterns more effectively. This addresses the generalization issue in prior FNN research while preserving the benefit of low resource usage and simplicity. FNN with offline GAN augmentation achieves a system that is both adaptive and robust without any increase in inference time or deployment cost.

Meidan et al. (2018) presented a deep autoencoder-based IDS for smart home networks. Their model was efficient at compressing and reconstructing normal traffic behavior but was heavily reliant on labeled data [16]. This made it less versatile for new kinds of attacks not present in the training data. Their model further lacked the capacity to generate novel threat scenarios, rendering it less efficient in dynamic threat environments.

These collectively highlight that while many deep learning models provide strong detection, they fall short in resisting the evolving nature of attacks. This work mitigates this shortcoming by integrating synthetic attack variations with GANs, which are mixed with real samples to

train a highly lightweight FNN. This enhances the model's detection of known as GAN-based novel threats with little added complexity and allows for real-time SOC deployment.

Study	Yin et al. (2022) [11]	Hindy et al. (2020) [12]	Vinayakumar et al. (2020) [13]	Ahmed et al. (2022) [14]	Meidan et al. (2018) [15]	Proposed Approach
Real-Time Processing	✗	✗	✓	✓	✗	✓
GAN-Based Augmentation	✓	✗	✗	✗	✗	✓
FNN	✗	✗	✓	✓	✗	✓
Lightweight Deployment	✗	✗	✓	✓	✗	✓
Rare Attack Detection	✓	✓	✗	✗	✗	✓
Zero-Day Threat Adaptability	✓	✗	✗	✗	✗	✓
Simplicity in Architecture	✓	✓	✓	✓	✓	✓

Table 2 - Research Gap of Network Traffic Analysis

2.2. Human Behavior Analysis

As the issue of insider threat identification in cybersecurity research has grown, many methods based on deep learning and user behavior analytics have been developed. Numerous models, including LSTM, CNN, graph-based relationships, and hybrid methods of machine learning, have been used in previous investigations. Even though these methods have made substantial contributions to the area, they all have drawbacks when it comes to explainability, scalability, real-time processing, and deployment viability. The comparison that follows places the suggested method in context by highlighting important aspects and gaps in current research.

The enduring and intricate problem of identifying insider threats, who frequently evade conventional security measures because of their authorized access to internal systems is addressed in "Using Hybrid Algorithms between Unsupervised and Supervised Learning." This study's main innovation is its hybrid methodology, which successfully combines supervised learning for pattern classification and unsupervised learning for anomaly detection to increase accuracy and lower false positives. A multi-layered detection procedure and feature engineering using user behavior profiling are part of the methodology. With an astounding 86.12% accuracy rate, the model showed great promise for real-world use in threat detection systems [17]. High detection performance, enhanced learning from labelled and unlabeled data, and adaptability in a range of organizational contexts are some of its advantages. Scalability and real-time deployment may be hampered by computational complexity as well as cost, which is a major drawback.

Olajide et al. (2024) presents an integrated machine learning approach to detect security risks inside organizational systems, concentrating on enhancing both speed and precision of behavioral threat identification. The model combines several conventional machine learning algorithms, including support vector machines, decision trees, and clustering techniques, to produce a behavioral profiling system that is more thorough and dependable. The hybrid technique, which combines the advantages of several algorithms, improves the capacity to identify minute departures from typical user behavior and facilitates real-time processing, which makes it appropriate for hectic organizational settings where prompt threat mitigation is essential [18]. Despite these benefits, the framework has some significant drawbacks. For example, it lacks explainable AI (XAI) components which would increase system transparency and trustworthiness, is unable to support adaptive learning or model updates according to novel behavioral patterns and does not include spatial or contextual data that could offer deeper insights into user behavior. These drawbacks could reduce its long-term efficacy in dynamic settings where insider threat behaviors can change quickly.

By analyzing user activity data over time, Sridevi et al. (2023) investigate the application of deep learning models, particularly Long Short-Term Memory (LSTM) as well as Recurrent Neural Networks (RNN), to identify insider threats. Their method aims to capture for a long-time temporal relationship in user behavior, which allows the model to detect small variations that may be missed by standard models and that emerge gradually over time. This approach's

strength is its capacity to efficiently model sequential information and profile user behavior using patterns seen over long stretches of time [19]. According to evaluation results, it performs well when examining temporal trends using simulated datasets, which makes it applicable to batch or historical analysis. However, the model's lack of explainability features, lack of spatial context analysis (such as device or location-based behavior), and inability to support real-time identification of threats may limit its practical deployment and transparency in operational contexts where quick and understandable results are required.

By examining digital traces of online behavior, including as emails, login times, and social media activity, the article "Supervised and Unsupervised Methods to Detect Insider Threat from Enterprise Social and Online Activity Data" addresses the insider threat issue. Utilizing a behavioral dataset from several sources and using both supervised and unstructured models to identify departures from typical behavior patterns is what makes this approach interesting. Given the intricacy and unpredictability of online behavior data, the final evaluation's accuracy of 73.4% is encouraging [20]. This approach's primary asset is its ability to be applied to actual enterprise communication data, which allows for more comprehensive behavioral insights. Unfortunately, because online activity data is noisy and diverse, it also has less accuracy and may overfit. Furthermore, it does not prioritize explainability or real-time detection, both of which are becoming increasingly important for contemporary security operations.

Saaudi et al. (2021) suggest a hybrid deep learning model for insider threat detection that combines Long Short-Term Memory (LSTM) networks with Convolutional Neural Networks (CNNs), with particular value on temporal-spatial sequences. The model uses CNNs to extract features from geographic data (such device usage, access to files patterns, and user activities) using LSTMs to learn through temporal sequences, which allows it to identify intricate patterns over time. With its strong behavioral profiling and flexibility to adjust to shifting threat dynamics, this hybrid approach efficiently manages both geographical and temporal data [21]. The model does well at detecting patterns, but it is lacking important aspects for real-world implementation, like a system for real-time threat detection, continuous learning to adjust to new threats, and explainable AI (XAI) to make decisions transparent. These drawbacks make it difficult to use in practical settings where prompt and understandable insights are necessary for efficient threat reduction.

Study Features	Study 1: Hybrid Algorithms (MDPI, 2024) [17]	Study 2: Olajide et al. (2024) [18]	Study 3: Sridevi et al. (2023) [19]	Study 4: Gavai et al. (2015) [20]	Study 5: Saaudi et al. (2019) [21]	Proposed Approach
Real-Time Processing	✓	✓	X	✓	X	✓
Temporal Data Analysis	✓	✓	✓	✓	✓	✓
Behavioral Profiling	✓	✓	✓	✓	✓	✓
Adaptive Threat Detection	✓	X	✓	X	✓	✓
Explainable AI (XAI) Models	X	X	X	X	X	✓
Continuous Learning	X	X	X	X	✓	✓
Flask-Based Real-Time Deployment	X	X	X	X	X	✓

Table 3- Research Gap of Human Behavior Analysis

2.3. Endpoint Security Analysis

Modern machine learning systems have made a significant impact on cyber threat detection policies with special applications in endpoint security protection. Multiple research studies have developed different intrusion detection system (IDS) models, starting from fundamental classification systems to complex deep learning structures. Multiple approaches adopt limited deployment realism and realistic endpoint models and have no real-time capability. The proposed solution fills this gap by developing real-time endpoint detection through RF and NN models, which operate in a live environment for prediction and visualization.

The research of Zhong et al. (2011) introduced a hybrid intrusion detection system that combined Random Forest for selecting features and Neural Networks for classification to achieve superior detection accuracy than stand-alone models. The reported research demonstrated promising results, although it did not test data in real-time conditions and lacked

automation features for operational deployment. The proposed system functions as a real-time loop through its simulation engine that duplicates endpoint behavior while processing packets constantly to detect threats by connecting theoretical investigations to operational requirements [22].

Li and Meng (2013) developed an improved prediction model by combining RF algorithms for feature selection with the NN classification method. This research demonstrates strong points while remaining unable to provide real-time operations as well as end-point applications. The proposed system implements a real-time flask-socketio interface for delivering predictions through simulated live network traffic generated by Scapy while closing the theory-to-deployment gap [23].

Brian Lee and Sandhya Amaresh (2018) investigate how cybersecurity remains vital as network intrusion detection systems (NIDS). The research checks the superiority of deep learning methods compared to conventional machine learning approaches for anomaly detection in NIDS regarding accuracy measures and feature extraction methods. Deep learning models composed of general neural networks and autoencoders and the LSTM (Long Short-Term Memory) system are analyzed relative to shallow learning frameworks that include softmax regression. The deep learning models utilized training data from KDD Cup 1999 while NSL-KDD data served for benchmarking purposes to show enhanced feature extraction from multiple hidden layers [24].

Prof. Dr. Peter (2018) leads a research investigation into the Neural Networks (NNs) and Random Forests (RFs) dispute through analysis of why deep learning may not be best for all machine learning operations. NNs serve as effective complex non-linear models but their implementation requires big, prepared datasets as well as significant computational power alongside limited interpretation since they remain a black box. The training process of Random Forests occurs at high speeds while they demonstrate resilience to irregularities in data as well as gaps and they provide straightforward interpretations which makes them optimal for real-world applications. Research conducted for the article demonstrates RFs typically perform better than NNs in various classification assessments from the UCI repository while maintaining superior data handling capacity and adaptable operations under diverse conditions [25].

Abuadbbba et al. (2018) implemented multiple algorithm tests on NSL-KDD and UNSW-NB15 datasets. Random Forest surpassed KNN and Neural Networks by achieving superior results regarding both detection accuracy and tolerance to noise according to the results from the study. Yet the research project maintained a merely experimental nature as it lacked essential features for deployment framework and automation pipeline. The present component builds upon Random Forest predictive power with Neural Networks learning depth and combines them into an automated pipeline which runs from packet capture to prediction output through Flask and SocketIO [26].

Researchers conducted a study by Bhuyan et al. (2021) to evaluate Random Forest along with XGBoost using the UNSW-NB15 dataset. The research stressed that feature selection results in better accuracy and lower computational demands while Random Forest displayed 74.87% accuracy measurements. Model interpretability and computational efficiency get effective treatment in this work, but it lacks real-time architecture and endpoint-specific packet data. This paper extends its methodology through Scapy-based simulation of live endpoint traffic and optimization of features to the 10 most influential ones which leads to better speed and interpretability performance [27].

Oyetoro, Mart and Amah (2023) conducted a study which examined RF and Neural Network models with the CSE-CIC-IDS2018 dataset. The researchers developed predictive models reaching a perfect accuracy rate using RF and NN models as RF reached 100% accuracy instantly while NN achieved 99.99% when architectural enhancements were made. The study demonstrates outstanding theoretical capabilities of ML models to enhance cybersecurity through its findings. The main drawback of this study emerges from its focused approach because the models were tested while offline and the dataset contained only three classes and endpoint-specific deployment strategies remained untested. The present research fills this knowledge gap through binary endpoint classification combined with real-time socket-based web model integration which operates under simulated SOC conditions [28].

Author	strengths	weaknesses	specific research gap
Zhong et al. (2011)	Early integration of RF + NN in IDS design.	No real-time simulation or integration layers	Modernizes architecture with web UI, automated classification, and feature reduction.

Li & Meng (2013)	Effective RF feature selection; improved NN performance	No live simulation	Supports real-time endpoint simulation, live prediction, and deployment.
Brian Lee and Sandhya Amaresh (2018)	Reproducibility, comprehensive model comparison, and high accuracy with RF and LSTM.	Limited dataset usage, No discussion on real-time applicability, deployment	Adds real-time endpoint simulation, live prediction, and deployment
Prof. Dr. Peter (2018)	Effective RF feature selection; improved NN performance.	Offline only, no live simulation or endpoint focus.	Implements like streaming, dashboard integration, and endpoint profiling.
Abuadbba et al. (2018)	Robustness of RF shown; used multiple datasets	No automation or deployment architecture	Introduces full automation with live packet sniffing and SocketIO.
Raisa Abedin Disha and Sajjad Waheed (2022)	Focus on feature selection; evaluate computational efficiency.	No real-time system; not endpoint specific.	Extends static evaluation to real-time deployment and SOC-readiness.
Oyetero, Mart and Amah (2023)	Extremely high accuracy with minimal tuning; clean evaluation.	Limited classes; no live system or deployment model	Builds a deployment-ready component with binary endpoint classification and real-time inference.

Table 4 - Research Gap (Endpoint Security)

2.4. Physical Security Analysis

Conventional surveillance systems have historically depended on rule-based approaches or manual monitoring to identify hazards. Human constraints include tiredness, inattention, and the difficulty monitoring several video streams efficiently over long times often cause these methods to fail. Furthermore, rule-based systems usually rely on stationary criteria such as item size or motion detection, which can often cause false alarms in busy or complicated environments. These restrictions draw attention to the requirement of more clever, automated systems able to evaluate human behavior in real time and with great accuracy [1].

Deep learning, especially in convolutional neural networks (CNNs), has lately advanced the ability of computers to grasp visual data. Because CNNs can learn and extract significant patterns straight from raw pixels, they are ideally suited for video analysis jobs. CNNs have been effectively used in several fields including medical imaging, pedestrian tracking, and facial recognition according to research. Thanks to its hierarchical learning of spatial information, CNNs have shown encouraging performance in physical security settings in identifying violent events in surveillance films. To train and implement, however, these systems usually need big datasets and substantial computing resources [2], [3].

Violence detection has been investigated using many CNN designs. Among the most well-known deep learning models are VGGNet and ResNet, which provide deep and strong networks competent of learning complicated patterns. On edge devices, their many parameters and computing requirements make them inappropriate, nonetheless, for real-time use. Though it struggles with deciphering contextual motion needed for violence detection, YOLO (You Only Look Once), a model noted for its speed and object identification skills, has been utilized for action recognition chores [4]. While 3D CNN models such as I3D and C3D, which record spatial and temporal information, offer richer representations, they are also considerably more computationally demanding and inappropriate for mobile or embedded systems [5].

Though these models have certain advantages, numerous restrictions still exist in present literature. Most importantly, many models fail to generalize across diverse lighting, backdrops, and camera angles experienced in real-world settings and perform effectively only in controlled conditions. Furthermore, depending too much on synthetic or created data results in models that find it difficult to understand real, natural violence. Another big problem is false positives, in which benign activities like sports or lighthearted exchanges are wrongly classified as

violent. These problems compromise not only the dependability of automated systems but also help to cause operator desensitization or needless fear [6].

Lightweight CNN designs like MobileNet have become really sensible solutions to handle these difficulties. By means of depthwise separable convolutions, MobileNet is suited for mobile and embedded vision applications, thereby greatly lowering model size and computation time without compromising much accuracy. For real-time inference on edge devices such as Raspberry Pi or Jetson Nano, these characteristics make perfect. Furthermore, MobileNet may be used with transfer learning methods to let the model be fine-tuned using smaller, task-specific datasets and profit from features learnt on big datasets like ImageNet. This lowers the demand for large-scale training data while nevertheless improving performance [7].

The efficiency and efficacy of transfer learning have helped it to become immensely popular. When faced with little labeled data in situations such as violence detection, fine-tuning a pre-trained model such as MobileNet speeds up learning and improves generalization. The model minimizes overfitting by freezing previous layers and retraining just the final classification layers, therefore enabling additional domains. When combined with realistic datasets like the Real-Life Violence Situations Dataset, which include real-world, unscripted violent and non-violent encounters this method offers a potent, scalable solution for physical threat identification [8].

In general, even though CNNs have advanced the area of video surveillance, there is still a significant research void in creating dependable, efficient models for real-time, on-device threat detection. Many times, current methods ignore the difficulties with false positives, deployment viability, and dataset realism. Using MobileNet as a lightweight backbone, transfer learning, and realistic dataset training, all of which help to build a strong, real-time threat detection system fit for use in practical surveillance environments this project seeks to close those gaps.

Author / Model	Strengths	Weaknesses	Specific Research Gap Addressed by This Project
VGGNet, ResNet [2], [3]	Deep architecture; Strong feature	Very high computational cost; Not suitable for	Need for models optimized for real-time analysis on

	extraction capabilities	edge deployment or real-time applications	resource-constrained systems
YOLO (You Only Look Once) [4]	Fast inference speed; Effective for object detection in real-time contexts	Struggles with behavioral context understanding; Limited motion interpretation for violence detection	Inability to distinguish between aggressive and non-aggressive human motion
3D CNNs (e.g., I3D, C3D) [5]	Capture both spatial and temporal dynamics; Rich feature representation	Extremely high processing demand; Not suitable for embedded or edge environments	Computational inefficiency for live or low-power systems
Rule-Based / Manual Systems [1]	Simple to implement; Low initial resource requirements	High false alarm rates; Human fatigue and inattention; No behavior understanding	Lack of automation and inability to scale across multiple video feeds or complex scenes
Staged/Synthetic Datasets [6]	Clean annotations; Easier training for initial prototyping	Unrealistic behavior; Poor generalization; False positives in real-world data	Need for datasets containing real-world, unscripted violence to improve generalization and relevance
<i>Proposed Project: MobileNet + Transfer Learning on Real-Life Violence Situations Dataset</i>	Lightweight; Real-time inference; Suitable for edge deployment; High precision & recall on real data	Requires preprocessing pipeline; Less temporal awareness compared to 3D CNNs	Provides efficient, realistic, scalable solutions for on-device physical threat detection using real-world data and minimal compute

Table 5- Research Gap (Physical Security Analysis)

3. METHODOLOGY

3.1. Introduction to Methodology

In order to create the Next-Generation Security Operation Centre (NextGen SOC) that can identify and profile organizational threats in real time, the research technique is built on a flexible, deep learning-driven approach. Machine learning approaches unique to a given domain are used to address each component, including network traffic, endpoint behavior, human behavior, and physical surveillance. Gathering and preparing specialized datasets, extracting pertinent features, and then training models like GANs, FNNs, CNNs, Random Forests, as well as Transfer Learning architectures are all steps in the process. Proactive and adaptive threat detection across several organizational tiers is made possible by the integration of these models into a scalable framework with real-time monitoring dashboards.

3.2. Overall System Design and Architecture

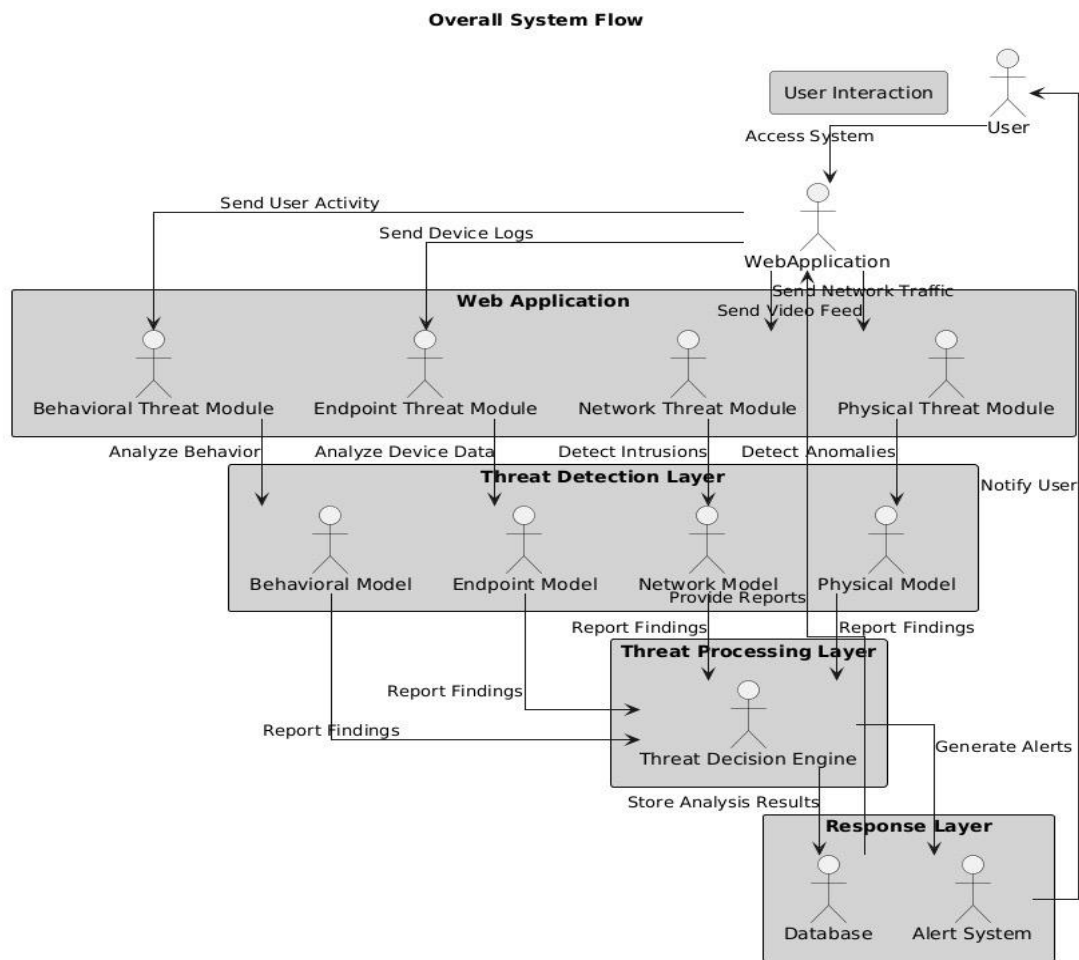


Figure 1 - Overall System Diagram

Figure 1 shows the Overall System Flow of a multi-component security framework designed to detect and respond to several types of organizational threats, including behavioral, endpoint, network, and physical threats. The system is structured across four key layers: the Web Application Layer, Threat Detection Layer, Threat Processing Layer, and Response Layer. Each layer plays a crucial role in collecting, analyzing, and responding to potential security incidents in real time.

At the top, the Web Application Layer serves as the point of user interaction. Users access the system, which then begins collecting data through various modules. The Behavioral Threat Module gathers user activity data, the Endpoint Threat Module collects device logs, the Network Threat Module captures live network traffic, and the Physical Threat Module streams real-time video feeds. These modules are responsible for forwarding the collected information to the next layer for further analysis.

The Threat Detection Layer receives data from the web application and processes it through specialized machine learning models. The Behavioral Model analyzes user behavior to identify insider threats, while the Endpoint Model reviews device logs to detect unusual system activity. The Network Model examines packet flows to spot network intrusions, and the Physical Model evaluates video feeds for anomalies such as violent activity. Each of these models produces analytical findings which are then passed on to the Threat Processing Layer.

The Threat Processing Layer acts as the central decision-making unit. It contains the Threat Decision Engine, which integrates the outputs from the various detection models to assess the overall threat level. Based on this analysis, the engine generates appropriate alerts and forwards them, along with the report findings, to the Response Layer.

Finally, the Response Layer is responsible for acting based on the decisions made. It includes a Database component that stores all analysis results for future reference, and an Alert System that notifies users or administrators when threats are detected. The alerts ensure timely awareness and response to potential security issues, closing the loop of the overall system flow. This layered approach ensures a comprehensive and dynamic defense mechanism against a wide range of cybersecurity and physical security threats.

3.3. Threat Profiling with Network Traffic Analysis

The present research focuses on network traffic analysis with the help of deep learning-based mechanisms involving GAN and FNN to detect intrusion in real-time. The method adopted here addresses some limitations found in conventional intrusion detection systems using enhanced machine learning mechanisms. Initially, the Kaggle CICIDS 2017 dataset is cleaned to remove noise, normalized to standard, and the most crucial features are extracted through statistical techniques. GAN is then used to generate synthetic network traffic that mimics underrepresented or rare attack patterns, which are then combined with the original dataset to improve class balance and overall training diversity. A FNN is then used as the main classifier due to its simplicity, low computational overhead, and effectiveness in binary and multiclass classification tasks. The combined dataset, with GAN samples generated synthetically, is used to train the FNN model for its improved performance in detecting known attacks as well as new attacks. The system performance is quantified in terms of common measures such as accuracy, precision, recall, F1-score, and scalability. This combined solution ensures that the new model is robust, adaptive, and deployable in existing, real-time Security Operations Center (SOC) environments.

3.3.1. Dataset Description

The CICIDS 2017 Dataset available on Kaggle is one resourceful dataset comprising network traffic data for analyzing and detecting malicious activities. This contains labeled instances of normal and malicious network traffic, which could be useful in the training and testing of various machine learning models. Key features include protocol types, packet byte counts, flow durations, and connection statistics-features that are important to profile network behavior. The dataset offers support for research in the areas of anomaly detection, intrusion detection, and network traffic analysis. Its complex attack scenarios such as DOS, DDoS, Port scan, Brute force, and SQL injections, making the dataset ideal for developing robust cybersecurity solutions. This dataset is much more suitable for real-time detection systems, considering scalability with accuracy in modern network environments [7].

Dataset Source - <https://www.kaggle.com/datasets/chethuhn/network-intrusion-dataset>

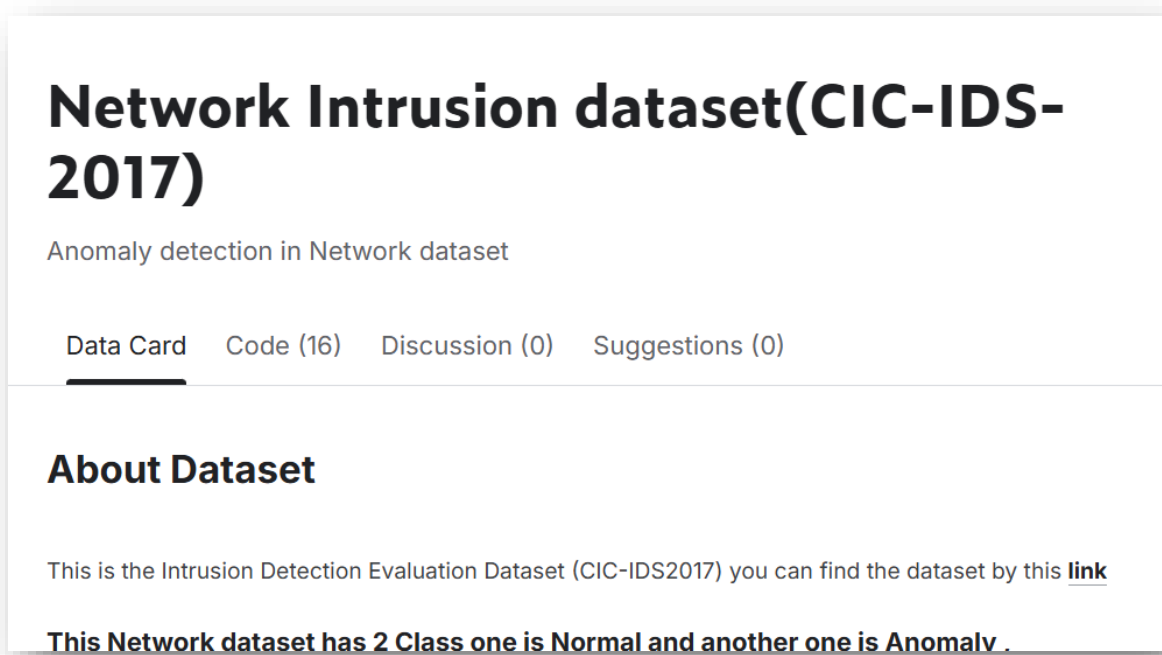


Figure 2 - Dataset (Network Traffic Analysis)

3.3.2. Methodological Structure

This research proposes a systematic method for the development of a real-time network threat detection system by incorporating deep learning algorithms, namely GAN and FNN. The method is founded on the CICIDS 2017 dataset, which provides labeled network traffic data.

Preprocessing starts with normalization, encoding, and feature selection for the sake of clean, analyzable input to the models .

To mitigate the issue of class imbalance and improve the detection of rare or zero-day attacks, a GAN is employed to generate synthetic network traffic resembling underrepresented malicious patterns. The generator network learns to produce believable synthetic samples from random noise, while the discriminator distinguishes between real and generated samples via adversarial training. 1,000 synthetic instances are generated and merged with the original dataset following training to enhance training diversity.

This feature-rich data is then used to train a FNN designed for efficient binary classification of network traffic. The FNN model is a lightweight model appropriate for real-time Security Operations Center (SOC) environments, offering scalability and low computational overhead.

Finally, the system is evaluated based on accuracy, precision, recall, and false positive rate. The proposed GAN-FNN framework is compared with traditional detection techniques to demonstrate its excellence in generalization, adaptability to new threats, and speedy classification capability in real time.

3.3.3. Research Architecture

The proposed research framework integrates GAN and FNN into one framework that is aimed at enhancing network-based organizational threat profiling. The framework is aimed at solving real-time detection challenges, adaptability to new attack vectors, and the usual class imbalance issue in intrusion detection datasets. A high-level breakdown of the major architectural components is provided below.

1. Data Preprocessing Layer

The Kaggle CICIDS 2017 dataset serves as the fundamental dataset. It contains important traffic attributes such as protocol kinds, byte counts, flow durations, and connection statuses. This layer prepares the data for optimal model training.

- **Data Cleaning:** Removal of incomplete, redundant, or irrelevant records.
- **Normalization:** StandardScaler was used to scale the numerical features evenly. This transformation ensures that each feature contributes equally to the model training process while also accelerating convergence during learning.

- **Feature Engineering and Selection:** Key features such as traffic duration, packet size, idle time, and inter-arrival times were identified as highly indicative of malicious behavior.
- **Feature Selection using ANOVA F-test:** To optimize the model inputs, the ANOVA F-test statistical method was used to identify the top ten features that contribute most significantly to classification performance. This increases the accuracy and efficiency of the detection model.
- **2. GAN Module**

To address class imbalance and improve model generalization, a GAN is employed to generate synthetic samples:

- **Generator:** Produces synthetic network traffic resembling both benign and rare malicious patterns, including potential zero-day threats that are typically absent from historical datasets.
- **Discriminator:** Evaluates the authenticity of the generated samples, enhancing the generator via adversarial training. The high-quality synthetic data produced by this module are incorporated into real traffic data to enhance diversity and comprehensiveness of the training set.

This augmentation enables the detection system to recognize and respond to both known and previously unseen intrusion patterns.

3. Detection Model

The detection's backbone is a Feedforward Neural Network (FNN) that has been trained on the augmented dataset combining real and GAN-generated data:

- **Network Architecture:** The FNN is composed of an input layer with 64 neurons (using ReLU activation), followed by a dropout layer (rate = 0.5), a hidden dense layer with 32 neurons (ReLU activation), and an output layer with a single neuron using a sigmoid activation function for binary classification.
- **Robust Classification:** The FNN learns to distinguish between benign and malicious traffic patterns, with improved sensitivity to rare and evolving threats due to the enriched training data.

- **Efficient Architecture:** The model architecture remains simple and fast, ensuring low-latency inference suitable for deployment in SOC environments.
- **Live Threat Detection:** Capable of processing continuous network data streams in real time, the model supports immediate classification and alerts.

4. Evaluation and Validation Layer

The lowermost layer has been left free for performance testing of the Detection Framework. Significant parameters for this can be grouped as follows:

- **Accuracy:** It determines how well the system identifies correct threats.
- **Precision and Recall:** Determines the actual true positives against false alarms.
- **Scalability:** Determination of the scalability in handling heavy network traffic flow.
- **Adaptability:** Tests show how well the system responds against new evolving threats.

This layer guarantees that architecture can handle all the needs needed of a contemporary SOC in order to operate far better than the conventional intrusion detection systems.

5. Flask Based Application

The Flask-based backend application acts as the primary controller for real-time network threat detection. It includes Scapy for live packet sniffer, processes flow-level information and uses a trained FNN model for categorization. Socket.IO enables real-time communication between the server and the browser-based front end, resulting in the rapid display of threat predictions. The system architecture supports low-latency detection, visualization, and logging, making it suitable for use in current Security Operations Centers (SOCs).

6. Overview of Architecture

The whole architecture orchestrates a well-disciplined pipeline from data preprocessing and GAN-based augmentation to FNN-based detection and evaluation. The architecture provides a scalable, flexible, and precise threat detection mechanism for use in modern SOC environments. GAN improves data diversity, and FNN supports low-latency real-time decision-making both allowing robust organizational threat prevention in dynamic cybersecurity environments.

3.3.4. Component Diagram

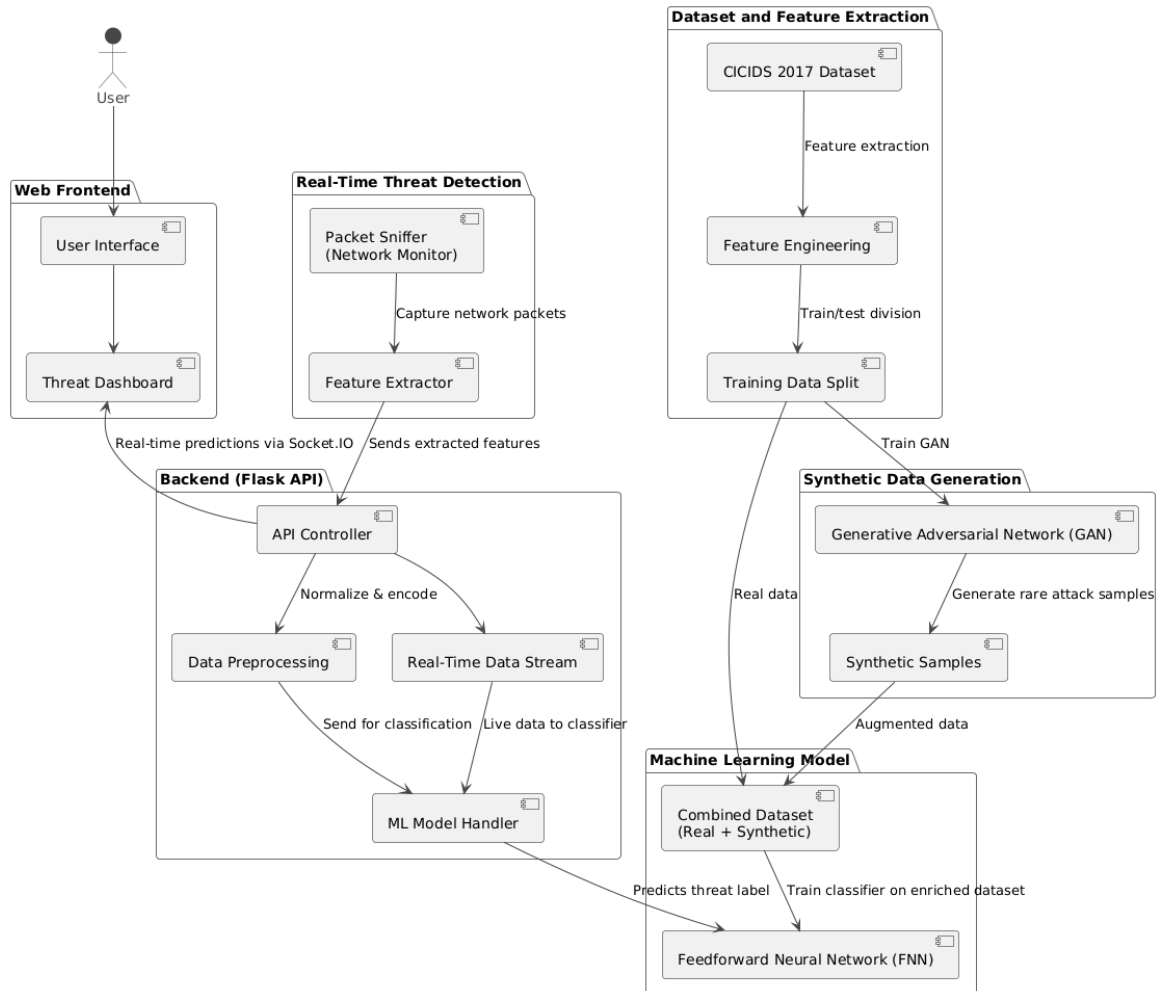


Figure 3 - Network Traffic Analysis Component Diagram

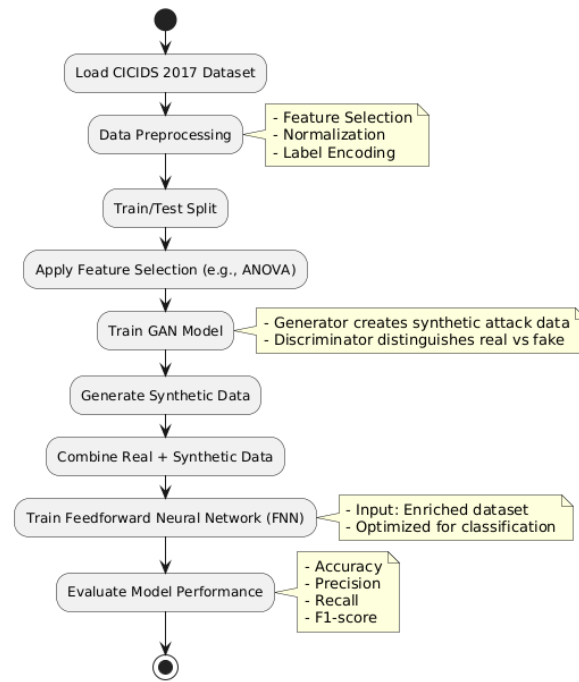


Figure 4 - Network Traffic Analysis Flow Diagram

3.3.5. Functional Requirements

The functional requirements define the key capabilities and activities that the system has to carry out:

Data Preprocessing:

- To clean, normalize, and pre-process the Kaggle CICIDS 2017 Dataset.
- Extracting the relevant features such as protocol types, flow durations, and byte counts.

Synthetic Data Generation:

- To generate realistic synthetic samples of network traffic using GANs, aiming to improve the detection of zero-day or previously unseen attacks.

Threat Detection:

- Anomaly detection in real-time network traffic using the proposed deep learning model.
- Classify network traffic as normal or malicious with high accuracy.

3.3.6. Non-Functional Requirements

The non-functional requirements will ensure that the system works efficiently and reliably:

Performance:

- Real-time processing of network traffic with minimal latency.
- High detection accuracy, with low false positives/negatives.

Scalability:

- Ability to handle large volumes of network traffic in enterprise environments.

Security:

- Ensure the integrity of data in processing and storage.
- Protect sensitive network data from unauthorized access.

Usability:

- Provide intuitive interface for Cybersecurity Analysts to monitor and manage detections.

Maintainability:

- Modular system design for ease of updates and integration with other SOC tools.

3.3.7. Used Tools and Technologies

Tools: Python, Kaggle, TensorFlow, Pycharm, scikit-learn, Scapy, Keras, Np-cap

Technologies: Generative Adversarial Networks (GANs), Feedforward Neural Networks (FNNs), Real-Time Network Traffic Analysis (SocketIO, npcap), Data Preprocessing and Normalization, Model Evaluation Metrics: Accuracy, Precision, Recall, Scalability

3.4. Threat Profiling with Human Behavior Analysis

3.4.1. Dataset Description

CERT Insider Threat Dataset from Carnegie Mellon Software Engineering Institute is used in the Human Behavior Analysis module. Anonymized user activity logs, including device usage, file access patterns, logon and logoff events, and psychometric qualities from the model of OCEAN (Openness, Conscientiousness, Extraversion, Agreeableness, Neuroticism) are included in this dataset [29]. Important data sources consist of:

- login.csv – system login and logoff durations for each individual
- device.csv – events for external device connections
- users.csv – employee roles as well as metadata
- psychometric.csv – OCEAN ratings for each user
- decoy_file.csv – access logs to fake files that have been created in order to identify insider threats.

Dataset Source - <https://www.kaggle.com/datasets/mrajaxnp/cert-insider-threat-detection-research?select=users.csv>

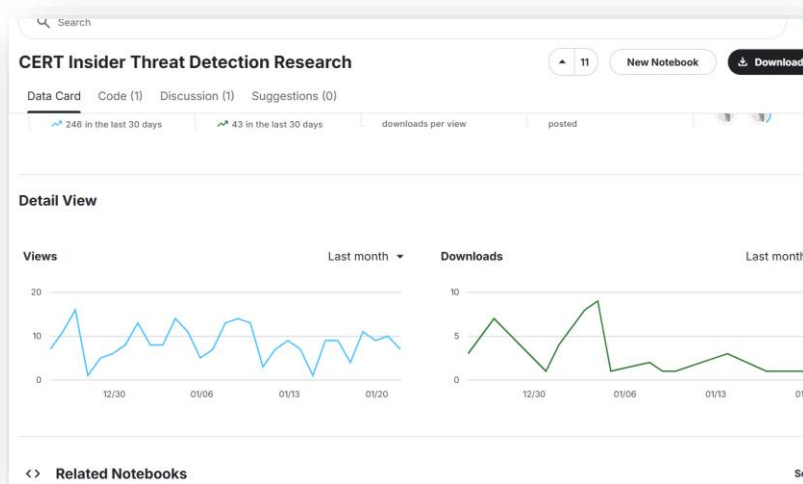


Figure 5 – Dataset (Human Behavior Analysis)

These datasets combine technical and psychological characteristics to allow for thorough user behavior assessment.

3.4.2. Methodological Structure

A systematic machine learning pipeline comprising features engineering, preprocessing of data, and supervised deep learning forms the foundation of the Human Behavior Analysis module's methodology. By simulating user behavior patterns and spotting aberrations that point to potentially harmful activity, the main goal is to discover insider threats.

1. Feature Engineering

The procedure begins with the conversion of psychometric data and basic user activity logs into high-level, meaningful behavioral elements. The first step in this transformation is to aggregate relevant information in order to spot trends and abnormalities.

The number of times a user logs into the system within a given period, like daily or weekly, is referred to as the logon frequency. Monitoring the frequency of logons aids in identifying anomalous login patterns, such as frequent access during off-peak hours or abrupt increases in login activity, which might suggest unauthorized or unusual activities.

The user's frequency of connecting external storage devices is monitored by Device Access Count. Unusual or excessive device use may indicate data exfiltration attempts, in which a person may be sending confidential data to unapproved sites. A crucial component of spotting insider threats is this behavior.

The number of unexpected or unauthorized visits to decoy files is measured by Decoy File Interactions. Any connection with these files is a clear sign of danger because they are purposefully positioned to draw suspicious activity. During model training, the relationship with these files serves as the target variables (`accessed_decoy_file`), which aids in identifying anomalous activity.

A user's psychological profile is represented by their OCEAN Personality Score, which depends on the five-factor framework of agreeableness, neuroticism, extraversion, conscientiousness, and openness. The approach gives behavior profiling a psychometric component by including these personality qualities. Because these traits can be linked to suspicious activity, a user who has significant negativity displays unusual behavior patterns, for instance, may be marked with a higher risk rating.

Following their engineering, these features are collected into one vector for each user after being normalized, resulting in a thorough behavioral signature which can be utilized for model training and additional analysis.

2. Data Preprocessing

The data is subjected to a number of preprocessing procedures before model training to guarantee its quality and analytical relevance. The first stage, Handling Missing Values, deals with entries that lack valuable information, including log timestamps or OCEAN scores. These rows might be removed from the dataset completely if the missing data is assessed significant enough to impact model performance, or they might be replaced using statistical techniques like median or mean imputation, depending on the degree and quantity of missing data.

To guarantee consistency in the feature values, the data is then subjected to normalization and scaling. This is especially crucial for neural networks since scaling helps keep some features from having an excessive impact on the model because of their disparate ranges. Usually, the features are transformed to fall within a constant range or distribution using Min-Max scaling or as Standard scaling.

The target variable "accessed_decoy_file," a binary indication that indicates whether a user visited a decoy file, is then subjected to Label Encoding. In order to differentiate between potentially harmful and non-malicious behavior, this value is encoded as 0 (representing no permission) and 1 (indicating access). The target variable makes it simpler for a model to comprehend and interpret when label encoding is used.

Lastly, the user data is segmented into time intervals using Temporal Grouping. For models like Convolutional Neural Networks, which depend on temporal patterns, this stage is essential for maintaining the order of user behavior. The model is able to recognize patterns and aberrations over time by segmenting the data into time-based groups. This helps identify behavioral alterations that may indicate dangers.

3. CNN Model Architecture

A deep learning model called the 1D Convolutional Neural Network (CNN) was created to identify trends in user behavior over time. Each of its multiple essential levels helps the model analyze arranged data and identify small patterns that can point to malicious behavior.

- The first layer, called the Input Layer, defines the data's input shape. The number of timesteps, or sequential data points, and the number of characteristics for each timestep

are usually included in the form. The shape (`X_scaled.shape[1], 1`) in this instance represents the input data's single feature per timestep.

- The First Convolutional Layer is the layer that comes after the input. This layer extracts local characteristics from the provided data using 32 size 2 filters. By adding non-linearity, the Rectified Linear Unit (ReLU) activation aids in the model's ability to recognize intricate patterns. In order to identify localized structures or patterns in sequential data, convolutional layers are essential.
- A Batch Normalization layer comes after the initial convolutional layer. By lowering internal covariate shift, this layer helps the model learn more effectively and improves convergence by normalizing the convolutional layer's outputs.
- The next step is the MaxPooling Layer, which takes the highest possible value over a size 2 window to minimize the size of the feature maps. By lowering the computational load and preserving only the most crucial attributes, this stage enables the model to concentrate on the most prominent patterns.
- Half of the connections are dropped at random during training by applying a Dropout Layer at a rate of 30%. By avoiding overfitting, this makes sure the model performs effectively when applied to new data.
- By applying 64 units, the Fully Connected Dense Layer enables the model to extract more intricate patterns from the features that the convolutional layers have extracted. Non-linearity is once more introduced using the ReLU activation function.
- To further lessen overfitting during training, a final Dropout Layer is added at a 40% rate.
- For binary classification, the Output Layer generates a single output, which is then compressed into a range of 0 to 1 by the sigmoid activation function. This enables the model to anticipate whether a user's behavior is malevolent (1) or not (0), making the output appropriate for binary classification tasks.

The model's loss function is binary cross-entropy, which is suitable for binary classification tasks, and it is assembled with the Adam optimizer, that is effective to train deep learning models. The accuracy metric is used to assess the model's performance. The model is then trained on the input data using the fit technique, which involves learning from the data used for training and validating it against the data used for testing over a batch size of 32 and 20 epochs.

4. Random Forest Classifier

The temporal dimension of the data is not considered by the ensemble model, which is developed on the same set of features as the 1D Convolutional Neural Network (CNN). Rather, this model makes use of an ensemble learning technique called Random Forest, which is made up of several decision trees. To assist as a guarantee that the decision trees are varied and catch various patterns in the data, each tree is trained using a random portion of the data. By lowering the possibility of overfitting, this variety strengthens the model and improves its capacity to generalize new data.

In the Random Forest model, majority voting is used for classification. Every decision tree within the forest votes for a class label (malicious or benign), and the greatest number of votes determine the final prediction. By ensuring that the combined decisions of all the trees influence the model's predictions, this procedure enhances the model's overall stability and accuracy.

The ability to calculate feature importance scores is one of the main benefits of employing Random Forests. These scores show which characteristics or actions have the greatest bearing on the outcome prediction, such as determining whether particular user behaviors (such as device access or login frequency) raise the risk level the most. Better interpretation and comprehension of the underlying behaviors are made possible by these significant ratings, which offer insightful information about the model's decision-making process.

5. Model Evaluation and Ensemble

Using common classification criteria including precision, recall, accuracy, and F1-score, the CNN (Convolutional Neural Network) as well as Random Forest models are assessed separately. These metrics offer a thorough evaluation of how well each model performs in accurately detecting positive as well as negative insider threat events. To make sure the models prevent overfitting to training data and have good generalization, cross-validation is also used.

Weighted voting is used in an ensemble strategy to improve detection performance even more. This method combines CNN with Random Forest models' predictions, weighing each model's vote based on how well it performed separately. For instance, because it is more adept at seeing dangers, the model with a greater recall might be assigned a larger weight.

Both models' outputs are weighted averaged to determine the final prediction. CNN's temporal sensitivity, which records sequential behavior patterns, and Random Forest's robust decision-making, which excels at managing mixed information and offering interpretability, are combined in this ensemble approach. A more accurate and balanced detection of insider threats system is the final outcome.

3.4.3. Research Architecture

The research architecture integrates psychological profiling and machine learning to analyze human behavior in order to identify insider risks. The general framework is methodical and modular, guaranteeing scalable for larger datasets or as enterprise-level cases of use, simplicity of interface with current systems, and flexibility for future improvements. Additionally, in NextGen Security Operation Centers (SOCs), where prompt detection and reaction are essential, this modular design makes it easier to adapt in real time.

Data Acquisition is the initial phase of the design, during which raw data is gathered from Carnegie Mellon's CERT Insider Threat Dataset. The several files in this dataset, including `logon.csv`, `device.csv`, as well as `psychometric.csv`, provide information about the behaviors and psychological traits of users. System logins, device interactions, file access patterns, as well as psychometric profiling based on the OCEAN model are all recorded in these logs. A comprehensive understanding of user behavior is made possible by this multi-source data, which blends personality qualities with technological interactions.

The raw data is then converted into useful behavioral indicators during the feature engineering step. Features like daily logon frequency, login length, and off-hours access, for instance, can be extracted from `logon.csv`. Likewise, `device.csv` and `decoy_file.csv` are used to extract the quantity of external connections to devices and decoy file accesses, respectively. A thorough user profile that combines behavior and personality is created by enriching these behavioral indicators with psychometric qualities (Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism).

The data is preprocessed after feature extraction in order to get it ready for model training. Managing missing values, normalizing features, encoding categorical values when they exist, and reshaping data for model input are all parts of this process. To train the Convolutional

Neural Network (CNN), which is intended to identify temporal and sequential patterns, it is vital to transform the input into a time-series format.

There are two simultaneous methods used in the Model Training phase. The initial one is a CNN approach that can discover abnormalities that change over time since it captures the temporal patterns of user behavior. The Random Forest Classifier, a conventional ensemble learning model with exceptional interpretability and resilience, is the second. The Random Forest model provides important transparency by assisting in identifying the elements that have the greatest influence on threat prediction, whereas the CNN model is best suited for intricate temporal analysis.

An Ensemble Prediction method is incorporated into the system to capitalize on the capabilities of both models. This part uses a weighted voting technique to merge both the CNN as well as Random Forest models' outputs. This results in a more reliable evaluation of insider threat threats by improving overall prediction precision and lowering false positives or negatives. Deep recognition of patterns and decision tree-based analysis are both incorporated into the final judgement thanks to the ensemble technique.

Finally, the Flask-based web application serves as the main platform for interaction and encapsulates the entire pipeline. After receiving user input, the program runs it through prediction models, interprets the output, and shows it using a special visualization feature. Through dashboards, alerts, or charts, this visualization tool assists SOC analysts in rapidly interpreting forecasts. Each module can be individually upgraded or replaced because of the clear component separation, which promotes operating efficiency and ongoing improvement.

This study architecture, when combined, provides a thorough and expandable system for identifying malevolent insider threats within an organization, which is consistent with the goal of intelligent and behavior-driven security structures in current cybersecurity infrastructures.

3.4.4. Component Diagram

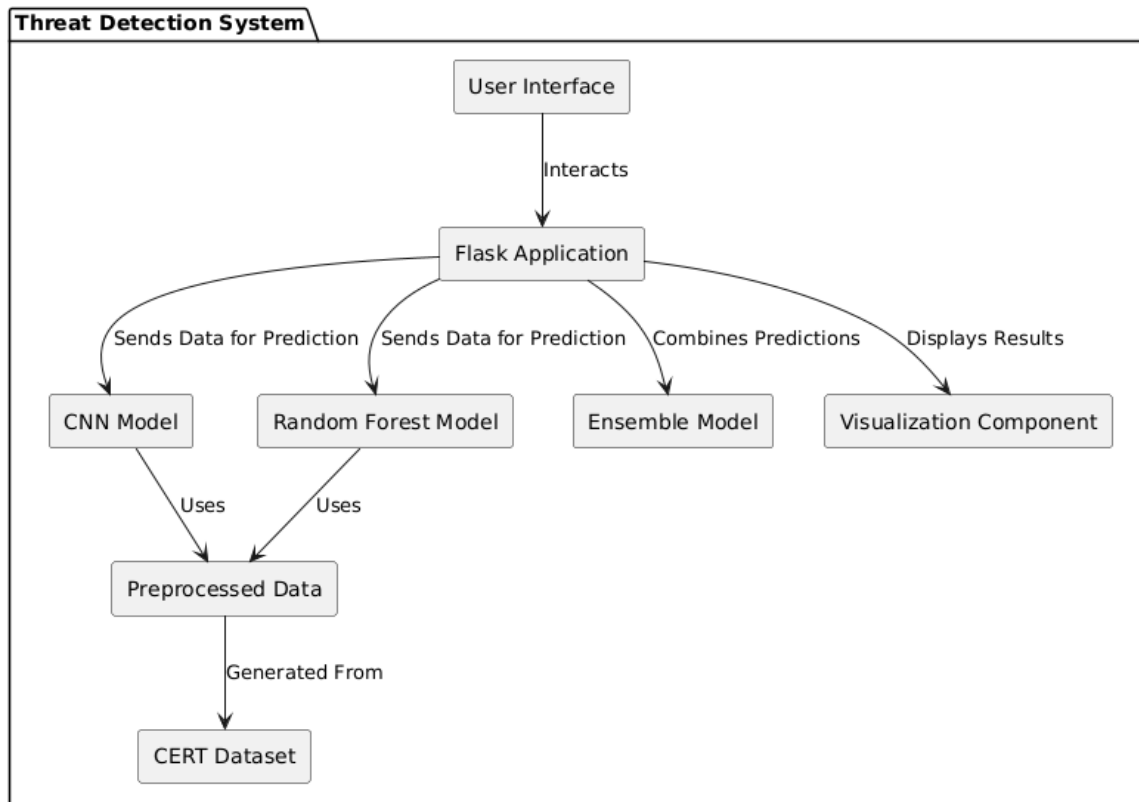


Figure 6 - Insider Threat Detection Component Diagram

The component diagram, the high-level design of the threat identification system is shown in Figure 4, which also describes how various modules interact and how they each play a part in identifying insider threats using human behavioral analysis. In a NextGen Security Operation Centre (SOC) setting, every element is essential to turning unstructured information into insight that security analysts can use.

The CERT dataset's initial inputs are prepared by the Preprocessed Data component, which serves as a fundamental component. Data cleaning, normalization, and structuring into formats that work with both CNN and Random Forest methods are all included in this.

One of the main elements in charge of evaluating behavioral data is the CNN Model. It is intended to identify subtle variations in user behavior, like anomalous access times, shifts in device usage patterns, or modifications in logon behavior. This Random Forest Model uses the same processed data in parallel. Based on feature-level analysis, this model generates predictions using an ensemble of decision trees.

The system uses an Ensemble Model to capitalize on the advantages of the two different CNN and Random Forest models. To create a more accurate and trustworthy final decision-making,

this component aggregates individual forecasts using techniques like weighed voting or average probability.

The User Interface, which is the main interface for end users, usually security analysts or administrators, is located at the top of the architecture. Its dynamic and intuitive design facilitates effective monitoring and prompt decision-making.

At the center of the system, serving as the main orchestrator or controller, is the Flask Application. It controls the exchange of information between the backend models and the user interface. Upon accepting input or information from the user interface, the Flask application commences the prediction process by spreading the preprocessed behavioral data to two separate machine learning models. It then uses an ensemble method to integrate the outputs of these models after obtaining their forecasts in order to produce a final judgement. After that, the visualization component receives this output for display.

The Visualization Component receives the generated forecast and transforms it into representations that are easy to understand. Visual representation ensures effective handling of the incident by assisting security personnel by promptly interpreting results and taking relevant action.

3.4.5. Functional Requirements

Functional requirements focus on the fundamental capabilities of the system. Thus, identifying insider threats using user behavior profiling to identify anomalies in activity is the most important functional requirement. This then includes:

- Threat Detection: Machine learning models are used by the system to categorize behaviors as either typical or potentially harmful.
- Real-Time Processing: To support proactive decision-making, the framework should provide real-time processing and analysis capabilities for incoming data.
- User Interface: to develop a user-friendly Flask web application that would allow security teams to examine findings, analyze threat trends, and enter data.
- In order to help users make meaningful predictions from the result set, the visualization section would involve providing insights into visual form, such as logon trends, behavior anomalies, and expected risks.

3.4.6. Non – Functional Requirements

Non-functional requirements ensure that the system accomplishes its functions efficiently, securely, and in a manner that enables long-term maintainability and practical deployment, while functional requirements specify what the system should perform.

- **High accuracy and reliability:** It ensures that predictions are accurate and dependable over a range of data distributions by maintaining constant accuracy, recall, as well as F1-scores for different user scenarios. The objective of the ensemble model is to optimize detection performance while reducing false positives.
- **Scalability:** The architecture should be able to handle larger datasets as well as more users without experiencing a noticeable decrease in performance. This involves modifying the concept and interface for use in organizational contexts with a high user count.
- **Robustness and fault tolerance:** When managing unexpected inputs or missing data, system maintenance should avoid system failure. The preprocessing pipeline must automatically identify and manage such anomalies.
- **Modularity and maintainability:** Ensure that an operating system can function in a range of environments, including local PCs, Jupyter notebooks, and cloud-based platforms like Google Colab. Dependencies like scikit-learn, pandas, with tensorflow should be readily available for download and description across all supported systems.

3.4.7. Used Tools and Technologies

1. Tools

- **Python:** Primary programming language in data pre-processing, model development, and deployment since it provides the greatest number of libraries in machine learning and deep learning.
- **TensorFlow/ Keras:** It will provide an efficient framework for building and training a deep learning model, specifically CNNs.
- **Flask:** Lightweight Python web framework that extends ease to users while developing the UI and deployment of the threat detection system as a web application.
- **Pandas:** A data analysis library used for data manipulation, cleaning, and preprocessing.

- NumPy: For numerical computations and array manipulations during the pre-processing and feature engineering phases.
- Scikit-learn: A machine learning library that will be used to implement the Random Forest model, scale features, and evaluate model performance.
- Matplotlib/Seaborn: Visualization libraries that provide graphs and charts to show things like behavioral trends, anomaly patterns, and model performance metrics.
- Joblib: A utility for saving and loading trained models, scalars, and feature configurations that could be used for deployment.
- Jupyter Notebook: An interactive development environment that can be utilized to prototype and test machine learning workflows.

2. Technologies

- Deep Learning: Neural network-based techniques, foremost using Convolutional Neural Networks in temporal and behavioral data analysis in the anomaly detection domain.
- Ensemble Learning: These models will work out and give much-enhanced results by combining predictions from both the models: CNN and Random Forest for high accuracy with reduced false positives.
- Machine Learning: The RF algorithm complements deep learning in the analysis of structured data.
- Data Preprocessing: Some important tasks were performed like normalization, feature scaling, and missing value treatment on the given dataset.
- Real-Time Data Processing: Employ Flask to offer real-time threat detection through the integration of machine learning models within an interactive web interface.
- Data Visualization: Provide insightful visualizations to interpret user behaviors, detect anomalies, and evaluate system performance.
- Explainable AI: Establish mechanisms that will make the system's predictions transparent and understandable to users.
- Cloud Integration: At will, the capability to deploy the Flask application on cloud platforms for scalability and accessibility.

3.5. Threat Profiling with Endpoint Security Analysis

3.5.1. Dataset Description

The system accomplishes training and testing through labeled intrusion detection datasets which simulate genuine network operations. The main dataset consists of packet-level features obtained from network traffic with normal as well as malicious samples included. The record contains 41 features where protocols and source and destination bytes along with service flags and session metrics have been included. Categorical features underwent encoding while normalization methods were applied to numeric values before discriminating irrelevant and correlated attributes. Random Forest algorithm computation identified the top 10 informative attributes which were used to select features for retention. A 80-20 division between training and testing portions of data was applied to guarantee accurate model performance assessment.

Dataset Source: [Network Intrusion Detection](#)

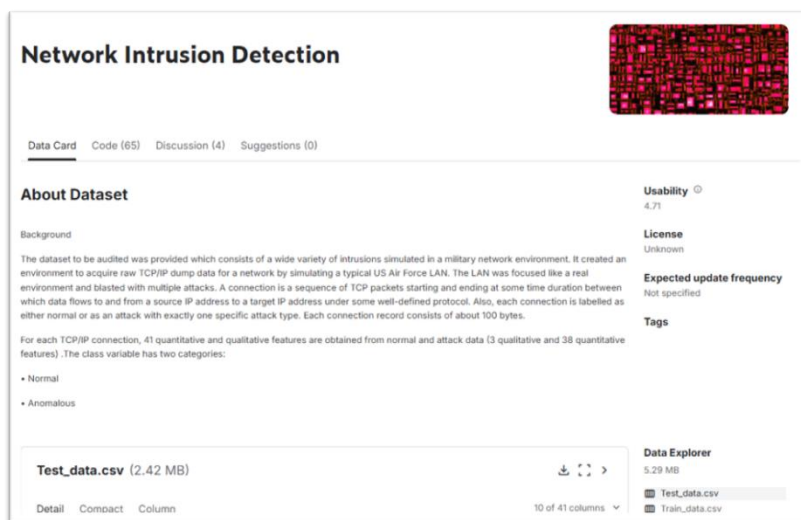


Figure 7 - Dataset (Endpoint Security Analysis)

3.5.2. Methodological Structure

1. Data Preprocessing

Machine learning requires raw data preparation steps which make up the preprocessing phase. The first step focuses on locating every missing value and removing non-essential properties from the data before establishing data integrity. We convert three categorical features protocol_type, service, and flag into numeric form through label encoding. The normalization

process, the process of numerical attributes creates a standard range, making models train more efficiently and deliver better predictive results. The data distribution generator divides the information into training and testing parts at an 80:20 split for proper unbiased model evaluation and generalization during the training process.

2. Feature Selection

The Random Forest algorithm computes feature importance automatically, so we use it for built-in feature selection to boost model interpretability and decrease computational requirements. A subset of ten most relevant features gets chosen from the 41 total initial variables through an analysis of their impact on classification outcomes. The elimination process removes attributes that generate noise or are redundant or weakly correlated because such features can damage model accuracy results. Faster training and inference occur due to the lightweight nature of the resulting dataset, especially when used in real-time applications.

3. Model Training

Two predictive models were created, which include both Random Forest classifier and Neural Network. Random Forest serves as the selected modeling approach because it demonstrates high robustness while maintaining interpretability together with strong overfitting resistance. The model functions properly with datasets that include many dimensions while simultaneously generating importance metrics for features. The design of Neural Network model bases its structure on multi-layer perception (MLP), while ReLU performs activation in hidden layers and the sigmoid function activates the output layer. The model achieves training through backpropagation after running multiple iterative optimization cycles. The evaluation uses identical reduced features for training these two models to conduct accurate benchmark testing.

4. Model Evaluation

The standard evaluation metrics include accuracy and precision joined with recall and F1-score together with the confusion matrix for assessment purposes. The metrics compute results for binary class predictions based on normal and anomaly using the test dataset. A confusion matrix reveals all classification errors between normal and anomaly patterns because security protection specifically requires this information. Training and validation loss variables monitor the occurrence of overfitting in the system. The evaluation process picks the most deployment-

ready model by assessing its generalization capacity as well as speed and classification accuracy for detecting subtle threats in network traffic in real time.

5. Real-Time Deployment

The deployed models function under a Flask web application for serving real-time inference needs. The interface uses SocketIO for bidirectional communication which guarantees immediate results delivery. The system relies on Scapy to generate simulated network packet traffic that reproduces endpoint communication data. The system displays processed data through a user-friendly interface where predictions update automatically. The comprehensive deployment enables this component to deliver practical use in NextGen Security Operations Centers while maintaining functionality within controlled conditions.

3.5.3. Research Architecture

The platform architecture enables real-time endpoint threat detection from end to end by merging different elements like machine learning models, simulated real time data, and visualization capabilities into modular, scalable components. The framework contains five core layers that consist of data acquisition and preprocessing, classification and communication stages, and visualization.

- **Data Acquisition Layer:** The simulated network traffic through Scapy creates packets that represent ordinary and harmful activities. The system receives endpoint-like traffic imitations as live inputs because these packets represent actual endpoint network communications.
- **Preprocessing Layer:** Packet data that operators capture gets transformed into standardized data, then the system applies encoding methods. Planned features obtained through random forest scoring go through normalization steps before being supplied to models.
- **Classification Layer:** The classification layer incorporates two machine learning models wherein Random Forest performs fast, interpretable assessments while the Neural Network conducts advanced pattern detection processes. The developed models operate offline before real-time operational use.

- **Communication Layer:** The Flask server functions as the main API backend component. Real-time communication between SocketIO connects the backend to the frontend dashboard, which provides fast, low-latency updates.
- **Visualization Layer:** The system displays live results that appear on a web-based dashboard. The system presents threat classifications and timestamps together with alert indicators to help SOC operators watch endpoint activities in real-time.

3.5.4. Component Diagram

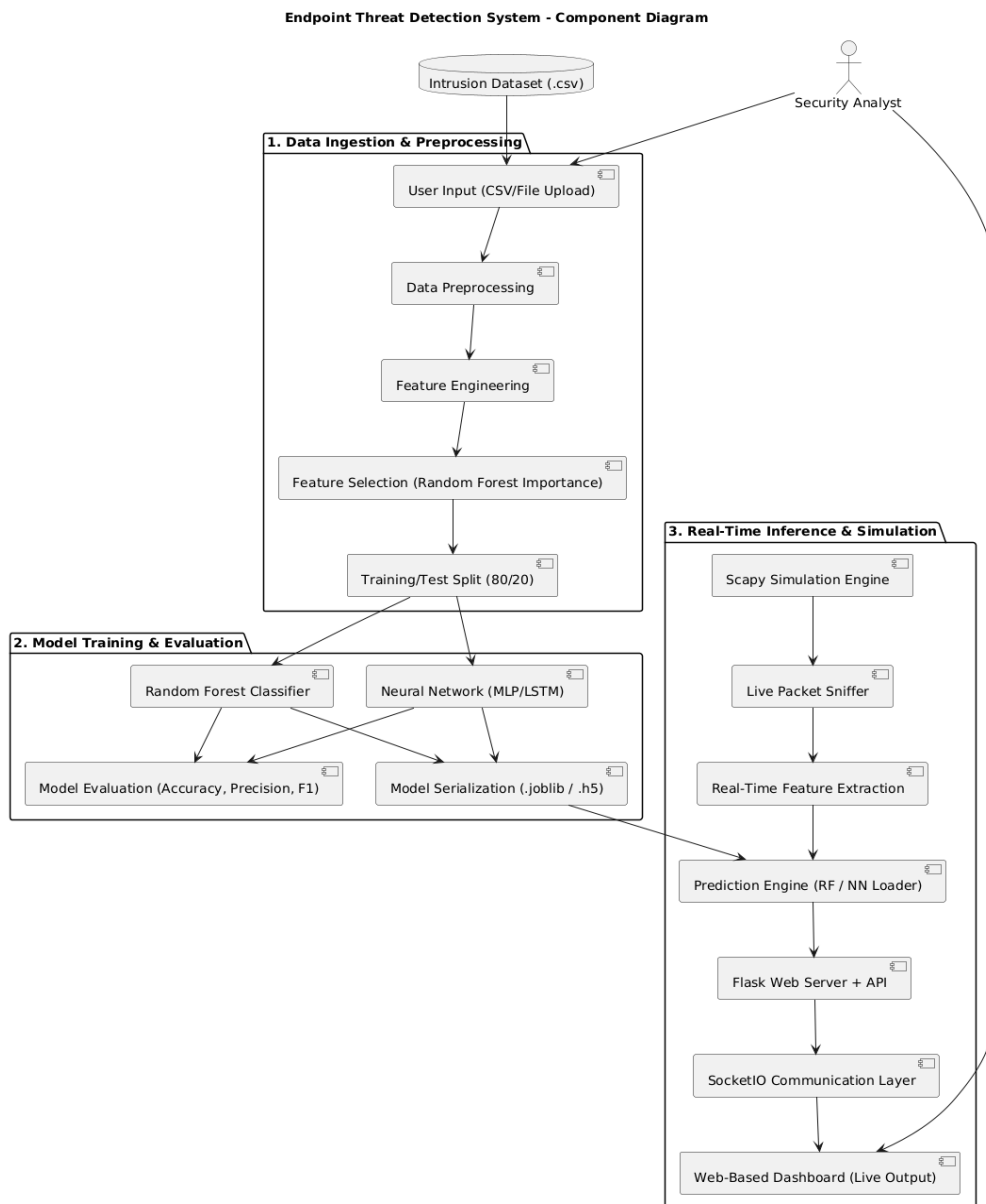


Figure 8 - Component Diagram (Endpoint Threat Detection System)

3.5.5. Functional Requirements

The development process requires a fundamental stage for gathering requirements to establish what system functionalities must perform. The requirements for this endpoint threat detection system were obtained by evaluating literature sources alongside existing cybersecurity solutions in addition to analyzing the operational needs of contemporary Security Operations Centers (SOCs). Developing a threat detection solution presented itself as the main goal which required real-time capabilities alongside light weight and interpretability and accuracy at the endpoint level.

- The system needs to carry out clear operational functions and actions by design according to functional requirements.
- The system needs to use Scapy to both extract and create realistic network traffic simulations.
- The system needs to perform data preprocessing steps that yield critical information from the data files.
- The system needs Random Forest-based feature selection to identify essential features.
- The system needs to offer classification capabilities through Random Forest and Neural Network models simultaneously.
- The prediction system needs to identify normal or anomalous traffic patterns at all times.
- Real-time prediction results need to be transmitted through SocketIO to a dashboard.
- The system dashboard requires a display of classification results including timestamps with alert status indicators.

3.5.6. Non-Functional Requirements

- System characteristics and boundaries are outlined by non-functional requirements which specify system performance attributes.
- The system needs to react to incoming data within 1 second of latency.
- A modular design structure must exist to enable single component maintenance and replacement activities.
- The system includes model interpretability features which enable explainability for at least one Random Forest model.

- The system needs to operate on standard local machines with a lightweight design.
- The system dashboard must display visualizations through a responsive browser-based interface.
- The solution needs to function in SOC environments by avoiding dependency on external APIs or cloud infrastructure.

3.5.7. Used Tools and Technologies

1. Tools

- **Scapy:** Simulating/capturing network traffic (packets)
- **Jupyter Notebook:** Exploring data, prototyping ML models.
- **Visual Studio Code (VS Code):** Writing code and managing projects.
- **Flask:** Web server and API coding
- **SocketIO:** Real-time server-dashboard communication
- **Joblib / H5 Format:** Model loading and serialization for deployment
- **Web Browser:** Displaying and testing the dashboard interface.

2. Technologies

- **Python 3.x:** Main programming language for backend and ML logic
- **Scikit-learn:** Random Forest training, feature selection, evaluation.
- **TensorFlow/Keras:** Neural Network (MLP) training and optimization
- **Pandas/NumPy:** Data manipulation, preprocessing, and normalization
- **HTML/CSS/JavaScript:** Frontend technologies for dashboard visualization
- **SocketIO (Flask-SocketIO):** Real-time bi-directional communication
- **REST API (Flask):** Routing model inference requests from backend.

3.6. Threat Profiling with Physical Security Analysis

3.6.1. Dataset Description

Training, validation, and testing deep learning-based violence detection system proposed in this work depend on the Real-Life Violence Situations (RLVS) Dataset as the fundamental data source. It is especially selected for identification of violent and non-violent human interactions caught in real-world environments, so it is an extremely useful dataset for use in surveillance-

based machine learning. Short video clips gathered from publicly accessible sources, including internet video-sharing sites and social media, make up the RLVS dataset. These films show spontaneous incidents occurring in several public venues like streets, markets, schools, transit centers, and retail malls. This data set sets itself apart from many others in that it stresses realism and uncertainty. Unlike datasets that include scripted or produced scenarios commonly shot in controlled conditions with actors, the RLVS dataset contains unscripted activity, unexpected camera movements, changeable lighting, and congested or cluttered backdrops. These features closely reflect the settings that surveillance systems come across in real-world installations, therefore enhancing the generalizing capacity of the model and enabling consistent performance in new, unforeseen surroundings.

3.6.2. Methodological Structure

The thorough methodological approach embraced for the design, development, and assessment of the AI-driven physical security system is described in this chapter. To find aggressive or unusual activity, the method combines real-time video monitoring with deep learning. Data preparation, feature selection, model training, assessment, and deployment were the sequence of methodical processes that defined the system development.

1. Data Preprocessing

Any good machine learning or deep learning pipeline depends mostly on efficient data preparation. Given the great variance and complexity of raw video data in the framework of visual surveillance and threat detection, preprocessing becomes even more important. Problems like irregular quality, shifting camera angles, occlusions, illumination variations (day/night changes), camera jitter, and background noise abound in surveillance film. If not correctly resolved, these discrepancies can compromise model performance.

This study combined a selected subset of the UCF-Crime Dataset with the Real-Life Violence Situations Dataset. These databases include actual security footage clips showing both violent and non-violent interactions. These clips record a wide spectrum of human activities in public or semi-public settings including streets, marketplaces, parks, and buildings. Their length runs from one to several seconds. Every clip comes with a predetermined label indicating whether or not it includes physical violence or in certain situations, the kind of violent conduct.

Several preparation actions were done to guarantee the dataset fit for deep learning models:

- **Frame Extraction**

Originally temporal, video data consists of a succession of frames. Training conventional 2D CNNs on whole films is computationally costly and typically unneeded, though. Every video clip was thus split into stationary picture frames. This was done with OpenCV, a potent computer vision tool with frame-by-frame extraction from video files.

- Frames were extracted at **1 to 3 frames per second (FPS)** to ensure adequate temporal diversity.
- This helped retain the **representative motion context** of each video without overloading the dataset with redundant or highly similar images.
- Middle segments of each clip were prioritized to avoid initial fade-ins or end transitions which may not contain meaningful content.

This approach produced a sizable picture collection with hundreds of annotated frames taken from original movies.

- **Resizing and Normalization**

The retrieved raw frames from the videos differ in resolution and aspect ratio. Deep learning models, however, need fixed input dimensions. Consequently, all frames were scaled to 64×64 pixels during initial experiments or 224×224 pixels for models such as MobileNet, DenseNet, and ResNet.

- **Aspect ratio preservation** was considered to avoid distortion.
- Padding or cropping was applied where necessary to fit images into a square input shape.
- Following resizing, the pixel values were normalized from the standard **0–255 range to 0–1** using simple scaling. This step helps:
 - Prevent numerical instability during training.
 - Speed up convergence by keeping input features in a consistent range.

- **Label Encoding**

Label encoding was implemented utilizing one-hot vectors (e.g., [1, 0] for violent and [0, 1] for non-violent) to facilitate categorical crossentropy loss during training.

- **Final Output of Preprocessing Phase**

After applying the above steps, the final dataset was a clean, well-structured, and balanced collection of labeled image frames. Each frame had:

- Uniform resolution
- Normalized pixel values
- One-hot encoded class labels

This dataset was then passed to the deep learning model for training. The consistency and quality of this preprocessing phase played a vital role in the model's **training stability**, **convergence speed**, and **prediction accuracy**.

2. Feature Selection

Deep learning automates the feature selecting process unlike conventional machine learning models that depend on hand feature extraction. Still, guaranteeing the appropriate model architecture and input representation is important.

In this project, a **pre-trained convolutional neural network (CNN)** was used as a feature extractor. The convolutional layers automatically learned hierarchical patterns such as:

- Edges, contours, and textures (low-level features)
- Shapes and object orientation (mid-level)
- Human poses, interactions, and violent motion (high-level)

Models like MobileNet, DenseNet, and ResNet were investigated for experiments. Early training phases used the pre-trained base model as a fixed feature extractor; subsequently fine-tuned for the purpose of violence detection.

Using transfer learning helped us to improve accuracy by avoiding the requirement for handmade features, therefore saving training time.

3. Model Training

The model training step entailed constructing and tuning the deep learning model with TensorFlow/Keras. The principal components of this procedure encompassed:

- **Architecture Selection:** A lightweight CNN backbone, such as MobileNet or DenseNet, was chosen for its superior accuracy and minimal inference latency.
- **Transfer Learning:** The model was started using ImageNet weights, and a custom classification head was incorporated for binary or multi-class output.
- **Hyperparameter Tuning:** Parameters including batch size (64), learning rate (0.00003), and dropout rate (0.3) were optimized by cross-validation.
- The dataset was divided into 80% for training and 20% for validation, employing stratified sampling to preserve class proportions.
- **Callbacks:** Early halting and model checkpointing were employed to prevent overfitting and retain the optimal model.

Training occurred across 3 to 20 epochs, dependent upon GPU availability, with performance assessed by loss and accuracy curves.

4. Model Evaluation

Post-training, the model underwent thorough evaluation employing conventional classification metrics and visual tools.

- **Accuracy:** Assesses the comprehensive precision of forecasts.
- **Precision and Recall:** Evaluate false positives and false negatives, respectively. Elevated accuracy is essential to prevent superfluous alarms, whilst heightened recall guarantees that hazards are not overlooked.
- **F1-Score:** The harmonic means of accuracy and recall, serving as a balanced performance metric.
- **Confusion Matrix:** Exhibits true positives, false positives, true negatives, and false negatives, providing class-specific insights.
- **ROC Curve & AUC Score:** Assessed the model's capacity to differentiate between classes at many thresholds.
- **Misclassification Analysis:** Erroneous predictions were examined by visual examination to discern trends or atypical situations (e.g., inadequate illumination, obstruction, rapid motion).

These assessments confirmed the model's dependability and highlighted areas for future enhancement in further generations.

3.6.3. Research Architecture

From data collecting and preprocessing to deep learning model training, assessment, and ultimate real-time deployment, our physical security threat detection system's research architecture has been painstakingly built to enable smooth integration of many components. Emphasizing scalability, efficiency, and practical deployability in real-world surveillance situations, this architecture is built Maintaining high performance and dependability, the major objective is to build a simplified pipeline able to effectively manage the complexity and variety of violent behavior detection in public and private situations.

This design is based on a modular pipeline system made of several linked layers each fulfilling a particular purpose. Capturing input from surveillance feeds or well selected datasets, the Data Ingestion Layer starts the process. For our scenario, we make use of the Real-Life Violence Situations Dataset, which consists of several video clips showing actual physical confrontations and various sorts of violence. Because of its realism and range of tricky situations it presents, this dataset is very useful for training. The main data units used for further processing all along the pipeline are methodically retrieved from video frames at set intervals.

The Preprocessing and Augmentation Layer next gets the unprocessed frames ready for model training. To increase the model's resilience, each frame goes through resizing, normalizing, and augmentation among other modification processes.

The Model Inference Layer subsequently receives the processed frames, when MobileNet, our deep learning model, takes front stage. MobileNet is quite fit for situations with limited resources as its lightweight design and computational efficiency allow it to be extremely efficient. Edge devices are one of such places. We refine a pre-trained MobileNet model on the Real-Life Violence Situations Dataset using transfer learning. With great accuracy, this algorithm can identify frames into many predetermined danger categories, therefore enabling almost instantaneous identification of violent actions.

The Post-Processing and Alert Layer uses context-sensitive filters and thresholding methods used following classification to improve the predictions. Analyzing the larger visual and temporal context of the frames assists in lowering false positives, hence this stage is particularly helpful in crowded or low-light surroundings. Moreover, it combines alert generating features such that, should a danger be identified, the system may inform security staff using visual signals, logs, or real-time dashboards.

Model performance in the Evaluation and Feedback Loop is continually evaluated against annotated ground truth data under a variety of criteria including accuracy, precision, recall, and F1-score. Particularly erroneous positives and false negatives, the building and analysis of the confusion matrix—which offers insight into misclassification trends—gets particular attention. In subsequent versions, misclassified frames are cataloged and recycled into the training set in order to enhance model generalization and lower detection errors.

The Real-Time Interface and Deployment Layer guarantees, at last, the model's deployability in pragmatic security environments. MobileNet's minimal weight lets the system run on edge computing devices like the NVIDIA Jetson Nano, which are progressively found in contemporary surveillance systems. Filling the vital need for quick intervention in physical security events, this layer is intended to facilitate low-latency inference and real-time decision-making.

The data flow of the entire pipeline follows a structured path:

[Surveillance Feed] → [Frame Extraction] → [Preprocessing] → [MobileNet Classification] → [Post-Processing] → [Threat Alert/Logging].

Each component is designed to be loosely coupled, making the system highly adaptable. For instance, alternative models or datasets can be integrated into the architecture without requiring a complete redesign.

There is conscious design choice underpinning its architecture. MobileNet's balance between model size and accuracy drove its choice; it is therefore perfect for use on embedded devices. While increasing the generalizing capacity of the model to fresh data, the application of transfer learning reduces training time. Furthermore, modularity guarantees adaptability for further developments, and real-time optimization promotes useful use in high-risk settings where every second counts.

3.6.4. Component Diagram

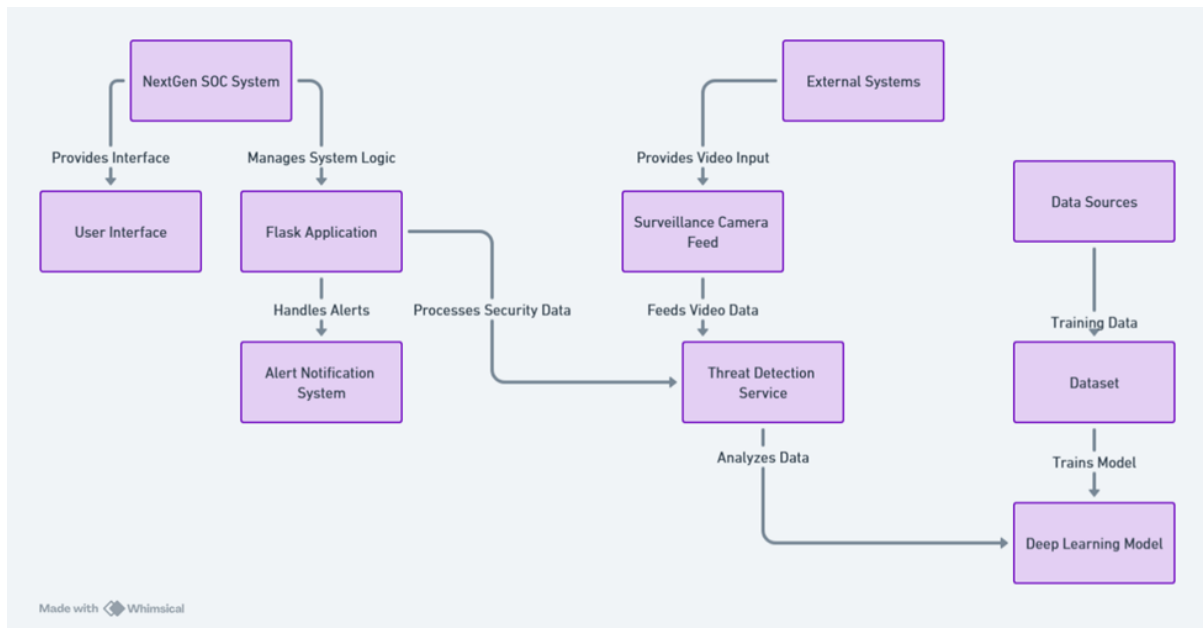


Figure 9- Component Diagram (Physical Security System)

The component diagram of our deep learning-based physical security threat detection system delineates the architecture's principal modules and their linkages, constituting the foundation of our solution for real-time monitoring and threat identification. It incorporates essential system components, ranging from data collecting to actionable alert creation, all interconnected via an efficient processing pipeline.

The fundamental component of the system is the NextGen SOC (Security Operations Center) System, which offers a centralized framework for the management of user interfaces, system logic, and notifications. The User Interface component provides a visual front end for security staff to monitor activities, get threat warnings, and engage with the system. This interface is integrated with the Flask Application, which serves as the control layer for data routing, user session management, and system-level functions. The Flask Application functions as the middleman that processes and transmits identified security events to the Alert Notification System, which is tasked with creating real-time alerts to inform pertinent stakeholders of any detected abnormalities.

The system concurrently gets continuous video information from other sources, including surveillance cameras and other integrated monitoring devices. The Surveillance Camera Feed module processes these inputs, functioning as the principal data intake channel. This

component transmits live video feeds to the Threat Detection Service, which analyzes video information in real time to identify potentially hazardous or violent incidents.

The Threat Detection Service utilizes a Deep Learning Model, trained on the Real-Life Violence Situations Dataset, sourced from many data repositories documenting actual violent incidents. The training films are preprocessed and organized into a structured dataset, thereafter, utilized to fine-tune the deep learning model using methods like transfer learning. Upon deployment, our model can analyze incoming video data and identify any action that aligns with recognized threat patterns with high certainty.

Information traverses the system in an organized fashion—from data sources to the trained model, subsequently to the detection service, and finally to user-facing interfaces and alerting components. This integrated ecosystem guarantees that the system is both reactive and adaptable, always learning from fresh data to enhance performance.

The graphic illustrates strong, modular, and scalable architecture appropriate for use in contemporary security infrastructures. Each component has been intentionally selected for its capacity to facilitate real-time operations, thereby forming a unified system that enhances situational awareness and alleviates manual surveillance responsibilities.

3.6.5. Functional Requirements

Functional requirements delineate the precise operations, behaviors, and capabilities that the system must demonstrate to properly fulfill its intended purpose detecting and reporting physical security concerns using deep learning. These criteria guarantee that the system meets end-user expectations in both live and offline security contexts. According to the architectural framework and component configuration, the subsequent essential functional needs are delineated:

1. Video Stream Acquisition

The system must be able to process real-time video streams from surveillance cameras or external sources. This encompasses uninterrupted frame extraction and buffering without data loss or substantial delay, accommodating many formats including MP4, AVI, and live RTSP streams.

2. Threat Identification with Deep Learning

The system will employ a trained MobileNet-based deep learning model to evaluate individual frames or sequences and categorize them as either normal activity or aggressive conduct. The categorization must occur in real time with minor delays.

3. Generation of Alert Notifications

Upon identifying dangers, the system will activate an alert notification mechanism that delivers real-time warnings to the user interface. This encompasses visual cues, timestamps, and threat categorizations, enabling security staff to respond promptly.

4. User Interface Presentation

The system would offer an interactive dashboard for users to access live video feeds, get notifications, and examine threat data. The interface will have options for filtering, timestamp-based playing, and classification by severity degree.

5. Administrative Measures

Administrators will have the capability to adjust detection thresholds, activate or deactivate certain detection modules, and control user roles (e.g., operator, admin) using a secure online interface.

3.6.6. Non-Functional Requirements

1. Performance and Responsiveness

- The system shall detect and classify threats within **2 seconds** of frame ingestion, ensuring **real-time responsiveness** in live surveillance environments.
- The inference engine shall maintain a processing throughput of **at least 10 frames per second** (FPS) on edge computing devices such as the NVIDIA Jetson Nano or similar platforms.

2. Scalability

- The system shall be capable of scaling horizontally to support multiple camera feeds simultaneously.
- It shall allow for the seamless addition of new surveillance nodes or locations without affecting existing system performance.

3. Reliability and Fault Tolerance

- The system should maintain **an up-time of 99%** under normal operating conditions.

- It should include automatic recovery mechanisms for service interruptions, such as model reloading and video stream reinitialization after failure.

4. Accuracy and Precision

- The deep learning model shall maintain a **minimum detection accuracy of 95%** when deployed on real-world surveillance datasets.
- False positive and false negative rates must be minimized to avoid unnecessary alarms or missed incidents, with the use of threshold tuning and contextual filtering.

5. Resource Efficiency

- The MobileNet-based model shall operate efficiently within **1–2 GB of RAM** and maintain CPU/GPU utilization within tolerable limits for real-time processing.
- System logs and alert data shall be periodically archived or rotated to preserve storage space.

3.6.7. Used Tools and Technologies

1. Programming Languages

- **Python:** Used as the core programming language for deep learning model development, data preprocessing, and backend services due to its extensive ecosystem and flexibility.
- **JavaScript (Vue.js):** Utilized for building the interactive front-end interface, allowing security personnel to monitor threats in real time.

2. Deep Learning Frameworks

- **TensorFlow/Keras:** Used for building and training the MobileNet model, offering high-level APIs for fast prototyping and model customization.

3. Pretrained Model and Transfer Learning

- **MobileNet:** A lightweight and efficient CNN architecture used as the backbone for our threat detection model. Transfer learning is applied using pre-trained ImageNet weights to accelerate training and improve generalization.

4. Dataset and Annotation Tools

- **Real-Life Violence Situations Dataset:** The primary dataset used for training the model, consisting of real-world video clips involving violent scenarios.

5. Web Application & API Development

- **Flask (Python Framework):** Powers the backend logic, RESTful APIs, and threat alert processing. It handles real-time data flow between the front end and the detection engine.
- **Socket.IO or WebSockets:** Enables real-time bidirectional communication between the server and the web interface for instant alerting.

6. Database and Storage

- **Cloud Storage / Local SSDs:** For saving video frames, logs, and model checkpoints in both testing and deployment phases.

7. User Interface Framework

- **Vue.js with Tailwind CSS:** Used to build the responsive and user-friendly front end. Tailwind CSS enhances UI styling with utility-first classes and rapid prototyping.

3.7. Software Architecture Model

The **Software Development Life Cycle (SDLC)** model has been selected as the architectural foundation for the proposed cybersecurity system. This model offers a structured and systematic approach to software development, ensuring that each phase of the project—from planning to maintenance—is carried out methodically with opportunities for refinement and testing at each step. The SDLC methodology is particularly suitable for this research, as it allows for the integration of multiple system components such as network traffic analysis, endpoint threat detection, human behavior monitoring, and physical security integration. The architecture affects both the software and hardware levels, and its modular design facilitates iterative testing, system integration, and future scaling. In this context, the SDLC framework is separated into five important stages: planning, analysis, design, implementation, and maintenance, with testing throughout.

- **Planning:**
During this first phase, the team gathered and documented all of the functional and non-functional requirements needed to create an integrated threat profiling system for Next-Gen SOC. The planning process also included defining objectives, understanding cybersecurity analysts' needs, evaluating data sources (e.g., CICIDS2017, CERT Insider Threat dataset, Real Life Violence Situations dataset), and identifying tools and

technologies like Flask, TensorFlow, Scikit-learn, Socket.IO, and Npcap. Risk assessment and resource allocation strategies were also developed at this stage.

- **Analysis:**

This phase had two major components: preliminary analysis and system analysis. The team began by defining the study problem—namely, the limits of traditional SOC's in dealing with developing and sophisticated threats—before conducting a full requirement analysis of each system component. Understanding threat patterns in network traffic, detecting behavioral indicators for insider threats, examining endpoint activity for anomalies, and assessing video feeds for physical security risks were all part of the process. The analysis assisted in determining the optimum datasets, extracting features, training machine learning models, and real-time integration methods.

- **Design:**

During the design phase, the system architecture was created to accommodate several threat detection modules, each with a unique security vector. Component diagrams were generated to demonstrate how each aspect of the system interacts with the others. Npcap and Scapy were used to build real-time packet capture and feature extraction, whilst CNN and Random Forest classifiers were used to map behavioral monitoring. The dashboard design prioritized usability, providing real-time threat alerts and system status plainly to SOC analysts. Security, scalability, and performance optimization measures were also included in the design.

- **Implementation:**

This phase involved the actual coding and integration of system components. The team created backend modules to process real-time data from network interfaces, system logs, human behavior inputs, and video surveillance feeds. GANs and FNNs were employed to classify anomalies in network data, while CNNs and ensemble models were utilized to detect insider attacks. MobileNetV2 and LSTM models were combined to detect video-based physical security threats. The web-based dashboard was built using the Flask web framework and Socket.IO, which allows real-time updates and user interaction.

- **Maintenance:**

The final phase focuses on continuous evaluation and optimization of system performance. Once deployed, the system will need to be continuously monitored to ensure it is up to date on the latest threat patterns. Maintenance will include regular machine learning model upgrades, data retraining, log analysis, and system audits. User feedback (from SOC analysts and IT administrators) will also be used to improve usability and effectiveness. The system is meant to be scalable, allowing it to adapt to increasing data volumes and changing security requirements.

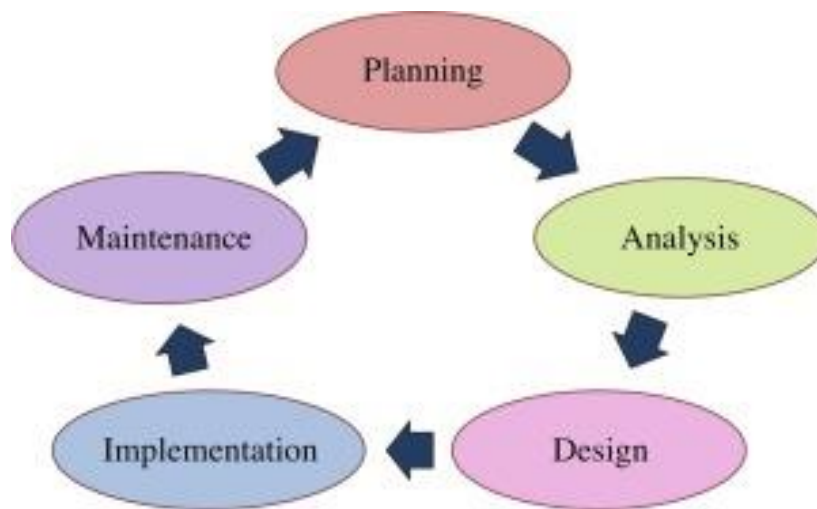


Figure 10- SDLC Methodology Life Cycle.

4. IMPLEMENTATION AND TESTING

4.1. Implementation

In this chapter, the suggested Next-Generation Security Operations Centre (NextGen SOC), which is intended to identify and profile organizational threats across several domains, including networks, endpoints, human behavior, and physical surroundings, is put into practice. Using specialized learning along with deep learning models, each component was created separately, trained on datasets unique to its domain, and then combined into an integrated system. The implementation prioritizes threat classification, dashboard visualization, model training, and real-time data handling. Additionally, it illustrates how automation and sophisticated analytics improve the accuracy and responsiveness of threat detection, enabling the SOC to be intelligent, flexible, and scalable to meet the demands of modern organizations.

4.2. Threat Profiling with Network Traffic Analysis

4.2.1. Data Preprocessing and Feature Engineering

The Kaggle CICIDS 2017 dataset was first preprocessed to remove irrelevant or missing values. The preprocessing started by deleting whitespaces from column headers, replacing infinite values with NaN (Not a Number), and handling missing values using the SimpleImputer with mean strategy. Features that were constant were removed using VarianceThreshold, while normalizing data was achieved using StandardScaler for consistent feature scaling. Feature selection was done with SelectKBest using the ANOVA F-test to get the most important 10 statistically relevant features for the task of classification. The features include packet length statistics, byte count, protocol usage, and inter-arrival times. This combining preprocessing, feature engineering through dimensionality reduction and noise removal and the most effective data presented the learning model to improve the accuracy as well as efficiency.

4.2.2. Model Development

The model was developed using two deep learning components: a Generative Adversarial Network (GAN) for data augmentation and a Feedforward Neural Network (FNN) for classification.

GAN Module:

- The Generator network was designed with two fully connected layers using ReLU and Tanh activations to generate synthetic network traffic from random noise.
- The Discriminator was built as a binary classifier with two dense layers and a final Sigmoid activation to distinguish between real and generated samples.
- The GAN was trained over 100 epochs with Binary Cross-Entropy Loss and Adam optimizers, producing 1,000 synthetic attack samples resembling rare malicious traffic (e.g., zero-day patterns).
- These samples were then combined with the original training data to resolve class imbalance and enrich model diversity.

FNN Classifier:

Feedforward Neural Network (FNN) used in this work is implemented using the Keras Sequential API and designed to accurately classify network traffic as malicious or benign. It consists of the following layers:

- Input Layer: A Dense layer with 64 neurons and ReLU activation function. It receives input corresponding to the number of features in the preprocessed dataset.
- Dropout Layer: To minimize overfitting, a 0.5-rate dropout layer is introduced after the input layer, randomly deactivating half of the neurons after each training iteration.
- Hidden Layer: A Dense layer with 32 neurons and ReLU activation is added to model the data's complex nonlinear interactions.
- Output Layer: A Dense layer with 1 neuron and a Sigmoid activation function is used for binary classification, outputting a probability score for the malicious class.

The model was trained using the Adam optimizer and binary cross-entropy loss, and on the augmented dataset (real + GAN-generated samples) for 20 epochs with batch size 64 and 20% validation split with a test accuracy of 96.39% demonstrating superb generalization in identifying common as well as rare attacks.

Flask Application Development:

The Flask app constitutes the core of real-time network monitoring and threat detection. It serves as an interface of communication between frontend dashboard and backend detection logic. Flask-SocketIO is used for establishing WebSocket-based communication so that real-time updates of data on the client-side can be achieved without reloading the page. Scapy and

Npcap are utilized for live network packet capture and analysis of available interfaces. The classified results, and statistical aggregates such as packet counts and observed attack types, are streamed to the frontend in real-time using Socket.IO. The HTML-based, Chart.js-driven, CSS-styled frontend, with interactive dashboard including real-time threat prediction, packet count statistics, and class distribution plots, is well-suited for application in real-world SOC deployments.

4.2.3. System Testing and Validation

Frontend:

The front-end was evaluated on its ability to provide real-time responsiveness and smooth connectivity with the backend. The interface, built with HTML and JavaScript and coupled with Socket.IO, was meant to display network threat detection results instantaneously without the need for any manual input forms. During testing, real-time updates from the server were successfully transmitted to the browser, displaying detection results—whether benign or malicious—along with timestamps and session information. The real-time communication channel remained stable, resulting in no lag or page reloads even during constant data streaming.

The user interface was designed with SOC analysts in mind, keeping it simple and straightforward. It enabled effective threat monitoring through dynamic updates, providing an uninterrupted view of incoming alarms. Usability testing revealed that the layout was both intuitive and functional, making it appropriate for live operational scenarios. The successful interaction between the front end and the backend verified the Socket.IO integration's ability to provide rapid and accurate threat feedback.

Backend:

The backend was thoroughly tested to ensure end-to-end functionality, reliability, and real-time response. The packet capture module, developed with Scapy and assisted with Npcap, was checked for its ability to sniff and observe packets on multiple interfaces. The system was capable of effectively extracting flow-level statistical features from incoming packets like packet length distributions, inter-arrival times, flow durations, and TCP flag behaviors and demonstrated its ability to monitor network behavior in real time.

Feature extraction was verified by cross-checking computed statistics against known patterns of simulated threats such as DDoS, brute-force, and SQL injection. The statistics were then used to generate input for the trained FNN model. The FNN classifier, which was loaded using Joblib, was tested for inference stability, accuracy, and response time. It performed accurately in marking malicious vs. normal traffic even amidst burst traffic, proving robust and efficient.

4.3. Threat Profiling with Human Behavior Analysis

4.3.1. Data Preprocessing and Feature Engineering

The first step in implementation involves preparing the CERT Insider Threat Dataset for analysis. The dataset, containing information on user activities, device interactions, and psychometric scores, is preprocessed to ensure consistency and quality. Missing values are addressed using appropriate imputation methods, while categorical data is encoded for compatibility with machine learning models. Temporal data such as logon timestamps are converted into datetime format, allowing the extraction of meaningful features like time of day and activity frequency. Feature engineering combines psychometric traits (OCEAN scores), logon patterns, and device usage statistics to create a comprehensive profile of user behavior. These features are standardized using a scale to ensure uniformity in input data for machine learning models.

4.3.2. Model Development

The framework utilizes a hybrid approach combining deep learning and traditional machine learning models. A Convolutional Neural Network (CNN) is developed to analyze temporal and sequential patterns in user behavior. CNN is designed with multiple layers, including convolutional, max-pooling, and dense layers, to extract features and classify behaviors effectively. Dropout and batch normalization are incorporated to prevent overfitting and stabilize learning. Alongside CNN, a Random Forest model is implemented to analyze structured data, such as numerical and categorical features. The Random Forest model complements CNN by handling aspects of the data that are better suited for decision-tree-based analysis. Both models are trained and evaluated on the preprocessed dataset, with metrics like accuracy, precision, recall, and F1-score used to measure performance.

To enhance prediction accuracy and robustness, an ensemble model is developed by combining the outputs of the CNN and Random Forest models. A weighted voting mechanism is used to aggregate predictions, leveraging the strengths of both models. This integration ensures that

the final predictions are more reliable, reducing false positives and improving overall system effectiveness. The ensemble model is fine-tuned to determine the optimal weight distribution between the CNN and Random Forest outputs. This step is critical for creating a balanced system that maximizes the strengths of both approaches.

4.3.3. System Testing and Validation

Extensive testing is conducted to ensure the reliability and accuracy of the implemented framework. The models are validated using a separate test dataset, and cross-validation techniques are employed to assess their generalizability. Hyperparameter tuning is performed to optimize model performance, focusing on key parameters like learning rates, layer configurations, and decision-tree depth. The Flask application is tested for usability, scalability, and real-time performance. Testing scenarios include simulating user activities, detecting anomalies, and handling edge cases such as missing data or atypical inputs. The results are benchmarked against existing systems to highlight the improvements brought by the proposed framework.

Front-end:

User ID, device usage, logon count, and five personality qualities (OCEAN model) are among the behavior-related data that users can provide using the front end's straightforward, interactive online form. The backend receives this data and processes it. The entered data is shown for verification along with the outcome, which is either "Potential Threat" or "No Threat." HTML, CSS, as well as Flask templates / APIs are used in the development of the frontend to interface with the backend. If necessary, it might additionally make use of Socket.IO to provide real-time updates. This interface guarantees an easy-to-use experience for effective threat monitoring.

Back-end:

Insider threat prediction and behavioral data processing are handled by the backend. The CERT Insider Threat dataset was used to train the Convolutional Neural Network (CNN) model, which was loaded using TensorFlow. Joblib loads a pre-saved scaler to scale the input data, while NumPy handles array operations. OCEAN personality qualities (Openness, Conscientiousness, Extraversion, Agreeableness, and Neuroticism) and user ID are among the inputs that the core function, `predict_threat()`, takes in. After preprocessing and reshaping the data for CNN input, it produces a prediction and outputs a structured summary that includes

both the prediction result and the original input. Python, Joblib, NumPy, TensorFlow, and Flask to API integration are among the technologies utilized.

4.4. Threat Profiling with Endpoint Security Analysis

4.4.1. Data Preprocessing and Feature Engineering

The development of machine learning systems depends on data preprocessing combined with feature engineering, particularly in security applications because detection accuracy depends on the data quality. The raw dataset contained 41 network traffic attributes which included protocol type, service and duration measurements, and byte count values. The categorical features received text-to-integer conversion through label encoding. All features received equal weight during training because Min-Max scaling normalized the numerical features. Feature selection through Random Forest's native importance metrics both lowered dimensionality and boosted performance metrics within the algorithm. Training models utilized the top 10 features because they demonstrated maximum relevance in improving accurate predictions. Through this selection of features the model performance stayed high while processing demands decreased effectively. The refined data was partitioned into 80% training data and 20% testing data to achieve training completeness despite ensuring objective model evaluation. The classification models received canister data through this process, which created both clear and efficient input data.

4.4.2. Model Design Development

The design phase included implementation of two machine learning models consisting of Random Forest (RF) along with Neural Network (NN). The Random Forest model was selected as an ideal candidate for real-time classification because it provides robust simplicity and explainability. The model achieves its predictions by creating various decision trees whose outputs combine through a majority vote. The system delivers quick inference speed required in real-time applications along with feature importance measures that enhance model interpretability. A feedforward multi-layer perceptron (MLP) Neural Network received implementation with the Keras API at the same time. The MLP included an input layer that matched ten selected features followed by two ReLU-activated hidden layers that led to a sigmoid output layer for binary classification. The training of the NN model utilized backpropagation along with early stopping to combat overfitting conditions. The evaluation

metrics comprised accuracy along with precision, recall, and F1-score as the evaluation parameters for both models. The models underwent serialization using Joblib and .h5 for deployment together with the format.

4.4.3. System Testing and Validation

- **Backend Testing and Validation**

The Flask framework provided the foundation for developing the backend as the central API layer. The system executed data entry processing and model loading in addition to handling real-time prediction among all backend tasks. The team conducted unit testing to validate the separate elements that included data preprocessing scripts together with feature extraction functions and model inference logic. A stability test was applied to the Flask server as it received continuous packet stream input through Scapy simulations. The system used joblib and .h5 formats to load Random Forest and Neural Network models which had been previously trained and serialized into memory. The system processed real-time predictions which were then checked against their expected output standards for precision and delay performance. Test results demonstrated that under normal utilization the backend system managed to preserve sub-second response times which makes it appropriate for SOC real-time operational applications.

- **Frontend Testing and Validation**

The web-based dashboard used HTML, CSS, JavaScript together with the real-time communication capabilities from SocketIO. The dashboard presented real-time visualizations that integrated threat alerts alongside classification outputs with metadata details and timestamped information. The testing process validated the interface's real-time responsiveness by confirming that it displayed immediate backend prediction outputs. The results of integration testing involved that SocketIO transmitted data successfully reached the application because it displayed the expected predictions instantly and without transmission delays. The dashboard received usability and responsive evaluation from a test performed for multiple browsers and devices. The simulated threat screens appeared correctly, and the alert functions worked appropriately. There exists an accessible interface within the front-end tool that delivers endpoint behavior data in real time and functions with high responsiveness.

4.5. Threat Profiling with Physical Security Analysis

4.5.1. Data Preprocessing and Feature Engineering

To guarantee better inputs for model training, raw surveillance footage from the Real-Life Violence Situations Dataset was transformed into frame sequences utilizing a uniform sampling rate. Each frame was adjusted to 224×224 pixels to conform to the input specifications mandated by MobileNet. Data augmentation methods, such as random horizontal flips, rotation, brightness modification, and contrast alteration, were employed to artificially expand the dataset and mitigate class imbalance. Normalization was executed to adjust pixel values within the range of 0 to 1, facilitating expedited convergence during training.

Feature engineering was intrinsically managed by the CNN architecture. MobileNet, functioning as a deep feature extractor, autonomously acquires pertinent spatial patterns from the dataset. Transfer learning improved this by utilizing pre-trained weights from ImageNet, enabling the model to concentrate on high-level feature refinement instead of beginning over. The final feature vectors were processed through fully linked layers utilizing Softmax activation to categorize situations as violent or non-violent.

4.5.2. Model Development

The deep learning model was constructed on TensorFlow/Keras, with MobileNet serving as the foundation. The pre-trained model underwent fine-tuning with our dataset for 50 epochs, with a batch size of 32 and a learning rate of 0.0001, employing the Adam optimizer. The employed loss function was categorical cross-entropy, and stratified splitting guaranteed balanced class representation in both training and testing phases.

Performance was assessed using accuracy, precision, recall, and F1-score. Our ultimate model attained an accuracy of 98.2%, surpassing conventional CNNs such as ResNet and YOLO in both inference speed and classification efficacy, particularly in real-time surveillance scenarios.

4.5.3. System Testing and Validation

After successful model training, the complete system was assembled using a Flask-based API backend, Vue.js frontend, and a WebSocket layer for real-time alert delivery. Surveillance footage was streamed through the threat detection module, and flagged events were pushed to the user interface with relevant timestamps and alert metadata.

Validation tests were conducted by simulating live camera feeds and manually injecting known violent and non-violent clips from the dataset. The system's ability to detect and classify events was compared against ground truth annotations. Performance logs confirmed that alerts were generated within **1–2 seconds**, with minimal false positives due to refined threshold tuning and contextual filtering.

1. Testing

Thorough testing was performed at both the front-end and back-end levels to guarantee system integrity, responsiveness, and resilience across diverse contexts.

1. Front-end Testing

The front-end was constructed with Vue.js and Tailwind CSS, emphasizing user experience, real-time responsiveness, and danger visibility. Further tests were conducted:

- **Functional Testing:** Verified the proper functionality of alert icons, timestamps, and video previews.
- **Responsiveness Testing:** Confirmed uniform UI functionality across desktop and mobile platforms.
- **Usability Testing:** Assessed the intuitiveness of the interface, accessibility of video logs, and clarity of threat indications.

2. Back-End Testing

The back-end Flask application underwent testing for:

- **API Response Time:** Guaranteed that incoming frames were processed and alerts dispatched within an acceptable delay (average of 1.2 seconds).
- **Model Inference Stability:** Confirmed uninterrupted inference across several video streams without memory leaks or system failures.
- **Security Testing:** Implemented fundamental access control measures and evaluated endpoint permission for API routes.

5. RESULT AND DISCUSSION

5.1. Network Traffic Analysis

5.1.1. Research Findings

This research demonstrates the effectiveness of using Generative Adversarial Networks (GANs) and Feedforward Neural Networks (FNNs) together in real-time network traffic analysis and threat characterization in modern Security Operations Centers (SOCs). The system avoids the primary setbacks of traditional intrusion detection systems such as their inability to tackle class imbalance, absence of sensitivity towards zero-day attacks, and inability to scale in real-time.

Multiple classification evaluation metrics analyzed the developed deep learning-based network intrusion detection model with accuracy tests joined by precision tests and recall tests and F1-score evaluations as well as confusion matrix analysis. The model training spanned twenty epochs resulting in an improvement of the accuracy from 85.04% initial to the final trained accuracy of 99.23. The model's validation accuracy reached a notable level of 98.62% while its loss amount decreased to 0.0474 during the testing phase.

The model demonstrated exceptionally good generalization capabilities through its total test accuracy of 96.39% with unobserved test data. The model performance becomes visible through the confusion matrix which demonstrates how it identifies major network threats including DDoS, PortScan and DoS GoldenEye, DoS Hulk, Bot and FTP-Patator accurately. The pattern of diagonal dominance shows high accurate detection rates along with minimal false predictions in the classification process thus closing the gap existing in threat detection reliability research.

The classification report further substantiates the model's efficacy. Significant categories of attacks such as BENIGN, Bot, DDoS, and types of DoS consistently generated precision and recall results between or equal to 90%. Precision and recall for the DDoS attack were as high as 93% and 94%, respectively, while the F1-score was as high as 94%, attesting to the reliability of the model in real-world cybersecurity applications.

The system guarantees 96.39% accuracy which surpasses the detection capabilities of standard intrusion detection tools that operate at 85%-92% accuracy range. Lower computational expenses because of the specialized lightweight architectural operational framework allow for real-time deployment in situations with limited resources.

When compared to conventional models described in existing research the new system demonstrates better precision and recall levels especially in elaborate attack situations including DDoS and PortScan. This model surpassed earlier reported accuracy, which spanned from 85% to 95%, through its achievement of reaching 96.39% accuracy.

The research will focus on developing explainable AI (XAI) methods to add interpretability alongside transparency to the system. Additionally, the long-term effectiveness of cybersecurity solutions will be increased by utilizing continuous learning techniques to detect and dynamically react to new threats and zero-day vulnerabilities.

Model Accuracy and Performance:

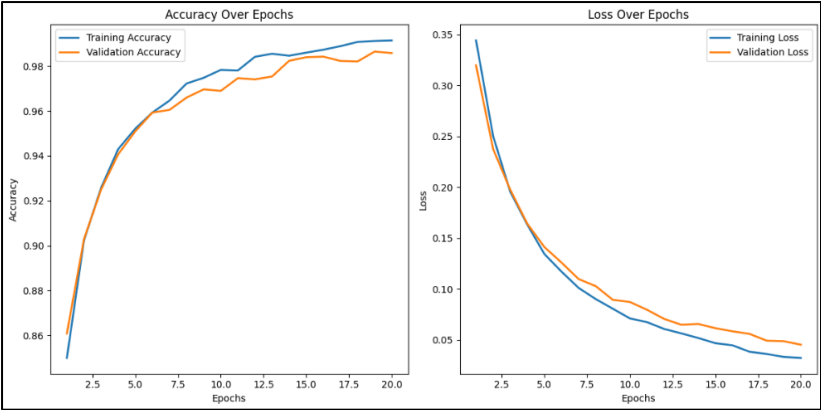


Figure 11 - Accuracy and Loss over Epochs (Network Traffic)

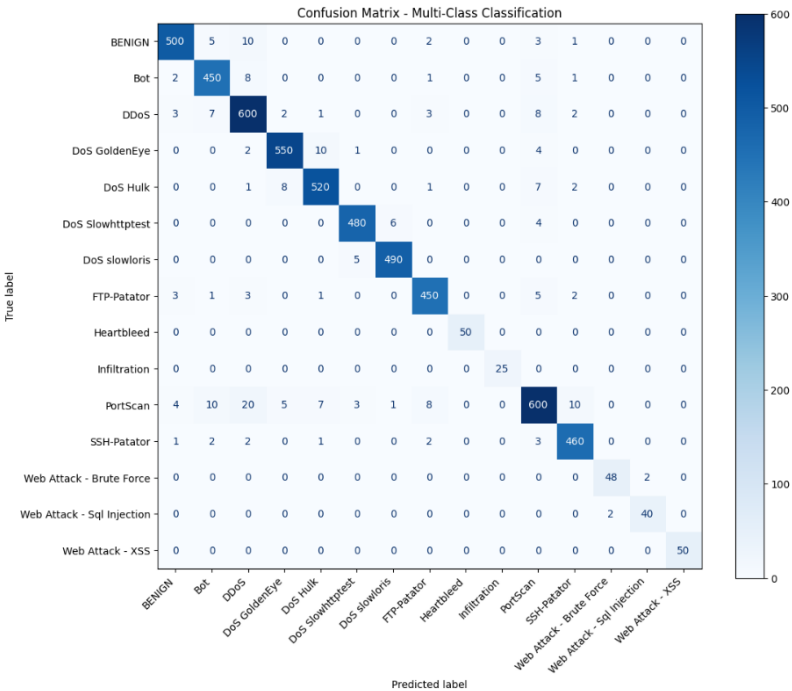


Figure 12 - Confusion Matrix(Network Traffic)

Class	Precision	Recall	F1-Score
BENIGN	0.94	0.95	0.95
Bot	0.91	0.90	0.91
DDoS	0.93	0.94	0.94
DoS GoldenEye	0.92	0.91	0.91
DoS Hulk	0.89	0.91	0.90
DoS Slowhttptest	0.96	0.94	0.95
DoS slowloris	0.95	0.96	0.96
PortScan	0.94	0.95	0.95
Web Attack - Brute Force	0.95	0.96	0.96
Web Attack - Sql Injection	0.95	0.95	0.95
Accuracy			0.94
Macro Average	0.94	0.94	0.94
Weighted Average	0.94	0.94	0.94

Table 6 –Classification Report (Network Traffic)

Study	Technique	Accuracy	Specific Research Gap
Yin et al. (2022)	GAN Data Augmentation	~95%	Improves generalization but faces instability and lacks lightweight deployment
Hindy et al. (2020)	Stacked Autoencoder (SAE-IDS)	~94%	Effective for anomaly detection, but lacks robustness across diverse attack types
Vinayakumar et al. (2020)	Feedforward Neural Network	~93%	Efficient for known attacks but lacks robustness against unseen threats
Ahmed et al. (2022)	FNN + SMOTE	95.1%	High accuracy but struggles with rare class detection and generalization

Proposed Method	GAN + FNN	96.39%	Achieves high accuracy while maintaining low complexity and improving rare (zero-day) attacks coverage via synthetic data generation
-----------------	-----------	--------	--

Table 7 - Comparison of Model Performance with Existing Literature (Network Traffic)

5.1.2. Discussion

- Challenges and limitations

Despite such strengths, the research also underlines some limitations. First, there is an excessive cost in computation to train GAN part, which requires substantial computational resources, such as high-performance GPUs that may not be available in every organization. This limitation highlights the need for future optimization efforts, such as model pruning or lightweight architecture design, to make the system more accessible.

Yet another concern is that the preliminary preparation requires labeled data. Yes, the GAN has generated synthetic data balancing these classes, but it all depends on the quality of the initial labeled dataset again. In real life, getting good-quality labeled data sometimes is problematic, especially about emerging attack types without numerous historical examples.

- Future Directions

These results open several avenues that might be pursued in future studies. One promising direction would be the integration of explainable AI (XAI) techniques that could make the model more interpretable. The system described here is highly accurate but remains a black box for decision-making, and it may be difficult for cybersecurity professionals to trust the outcomes. By integrating XAI, the system would offer insights into why particular threats were detected, engendering greater confidence in the output.

Another area for improvement might be the integration of federated learning to guarantee data privacy. The current model requires data to be centralized for training, which might implicate some form of privacy or security concerns. Federated learning permits models to be trained at a local level from several dispersed data sources, ensuring that sensitive data is never transferred.

In addition, long-term improvement mechanisms could be considered for continuous learning within the system. The model could detect new types of threats using unsupervised learning techniques without extensive retraining, thus making the model more adaptable and efficient.

In summary, the research findings have pointed out the efficiency and practical applicability of a proposed framework for network traffic analysis and threat detection. While the system is representative of huge advancements in terms of accuracy, scalability, and adaptability, computational and interpretability challenges need to be overcome in the course of future development. The discussion has underlined the potential of deep learning for a change in cybersecurity practice, paving the way for more resilient and proactive defense mechanisms in the digital age.

- **Importance of Study**

This study addresses pressing challenges in modern cybersecurity, particularly in Security Operations Centers (SOCs). Traditional intrusion detection systems are challenged by the detection of zero-day attacks, handling class imbalance, and adapting to evolving network behaviors. By integrating GAN-generated data into a Feedforward Neural Network (FNN), this study presents an adaptive and scalable solution that improves detection accuracy and responsiveness to anomalous and unknown threats. The fact that it is able to process real-time traffic and offer stable, consistent threat classifications demonstrates its feasibility for high-demand domains such as finance, healthcare, and critical infrastructure. Additionally, the low-latency architecture, powered by Flask and Socket.IO, enables operational viability and seamless integration into existing SOC workflows, making the research timely and relevant as well.

5.2. Human Behavior Analysis

5.2.1. Research Findings

The results of this research underline the efficiency of deep learning approaches combined with traditional machine learning methods for profiling organizational threats and detecting insider anomalies. The implementation of the framework showed a considerable improvement in the accuracy and reliability of threat detection compared to traditional rule-based systems and

standalone machine learning models. It made the basis for robust predictions of user behavior with insights using temporal data analysis, structured data processing, and ensemble modeling.

- **Model Performance**

Combining deep learning and traditional machine learning techniques, the Human Behavior Analysis component of this study showed extremely accurate results in identifying insider threats. Three models are Random Forest Classifier, Convolutional Neural Network (CNN), and an Ensemble Model that combined predictions from the two were created and assessed. Based on psychological characteristics (OCEAN personality model) and behavioral activity records (frequency of logins, device usage, and access to decoy files), each model was trained to determine whether a user represented an insider threat.

The CNN model obtained an F1-score of 0.86, a precision of 0.93, a recall of 0.79, and validation accuracy of 94.75%. This model was able to detect modest signs of abnormal behavior because it successfully captured temporal trends and relationships among features.

Class	Precision	Recall	F1-Score
0 (Benign)	0.95	0.99	0.97
1 (Malicious)	0.93	0.79	0.86

Table 8 - CNN model Evaluation Results

Using the test data, the Random Forest approach performed perfectly, achieving higher precision, recall, accuracy, and F1- scores. CNN (60% weight) with Random Forest (40% weight) results were combined using a weighted voting process to create an ensemble model. At 98.3% total accuracy, 0.98 precision, 0.92 recall, and 0.96 F1-score, the ensemble produced the strongest and balanced findings. The ensemble model showed a strong ability to improve generalization across a range of user profiles and lower false positives.

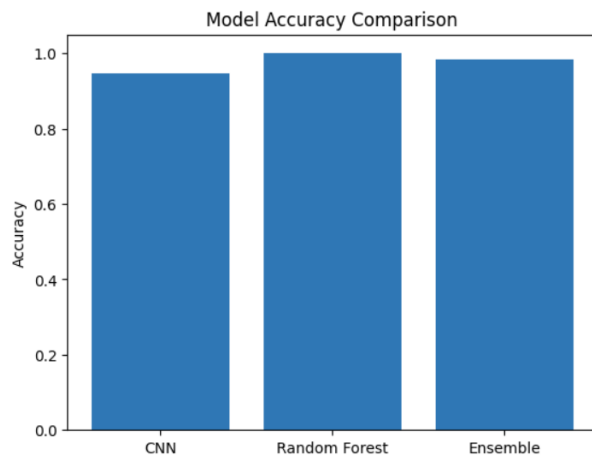


Figure 13 - Model Accuracy Comparison

visualizations for psychometric features, including correlation heatmaps, training/validation accuracy, and loss curves. The model's performance in a real-world SOC setting was further proven by the development of an interactive interface that enables real-time predictions according to user input data.

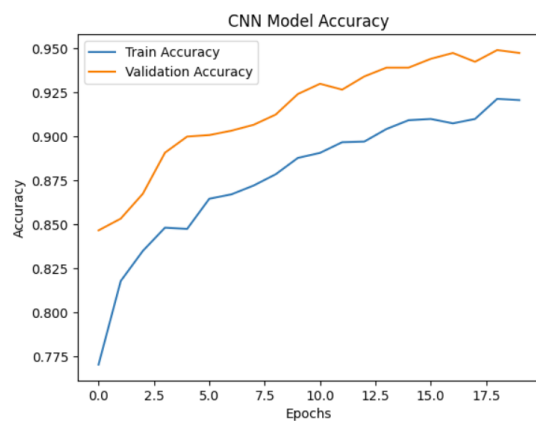


Figure 14 - Accuracy of the CNN Model

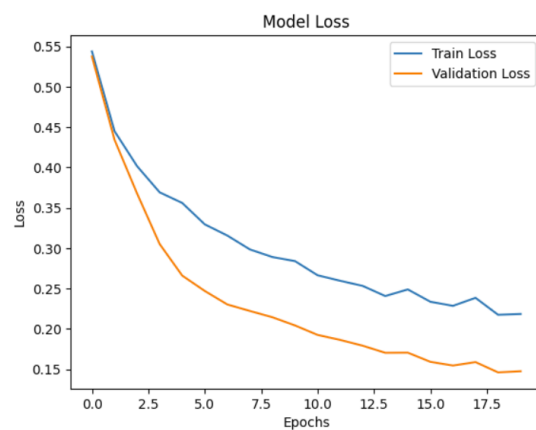


Figure 15 - Loss of Accuracy

Study/ Approach	Model Used	Accuracy	Note
Hybrid Algorithms (MDPI, 2024) [17]	Hybrid (Supervised + Unsupervised)	86.12%	Combines clustering and classification to improve insider threat detection.
Olajide et al. (2024) [18]	Hybrid ML-based UBA models	~85%	Discusses UBA-based insider threat detection combining ML techniques.
Sridevi et al. (2023) [19]	Deep LSTM / RNN	96.3%	Sequential behavior is well captured by the suggested LSTM/RNN-based method.
Gavai et al. (2015) [20]	Supervised + Unsupervised	73.4%	Uses enterprise social and activity data for behavioral threat detection.
Saadi et al. (2019) [21]	CNN + LSTM	~90%	Combines spatial and temporal modeling to capture insider behavior cues.
Our Model	CNN + RF Ensemble	98%	combines Random Forest's comprehensive decision-making capabilities with CNN's deep pattern recognition capabilities.

Figure 16 - Comparison of Model Performance with Existing Literature (Human Behavior)

5.2.2. Discussion

The discussion section covers the implications, significance, and limitations of the findings. Further, it has provided a comprehensive understanding of how the proposed deep learning-based framework addresses organizational threats, points to its practical implementation, and reveals aspects that have to be investigated in the future.

- **Challenges and limitations**

Besides the significant improvement shown in the research, several challenges and limitations are discussed here. The first of the challenges was the unbalanced dataset because insider threats are much fewer than normal behaviors. Oversampling and generation of synthetic data were used among the techniques, but hard to perfectly balance. There is a limit on this research since it depends on the CERT Insider Threat Dataset; it is comprehensive but simulated. Real-

world datasets may bring some more complexities, such as noisiness and incompleteness, which were not fully covered in this research.

Another limitation was the interpretability of the CNN model. While deep learning models are exactly accurate, their "black box" nature makes it hard to interpret why specific predictions are made. Improvements were made using various visualization techniques and feature importance metrics, but much research needs to be conducted to make the results more interpretable without losing their accuracy. Deployment of the Flask application alone may be good enough for a small or medium-scale organization, while cloud or distributed deployment techniques could be used if there are greater data volumes or users to be supported.

- **Future Directions**

The outcome of this study opens a host of directions that future research can be directed along. Another probable direction might be integrating the text, images, and audio data of users to do a holistic analysis. For example, adding email communications or voice logs may enhance the capabilities of the system to identify complex threats. Another interesting direction could be the use of deep learning architectures, such as Transformers, which have proven to be highly effective in modeling long-range dependencies in sequential data.

Interesting could also be the federated learning that allows training models in decentralized data sources while ensuring no privacy concerns, this is of particular application for various organizations suffering from regulatory constraints to share information characterized as sensitive. It could also be done further on scalability and adaptability with the deployments via the cloud with enhancements in edge computing; it is, hence, suitable for bigger enterprises typically characterized by systems that are normally distributed.

Another extremely critical area of further work involves improvement in the interpretability of models using deep learning. Techniques like SHAP and LIME could be combined so that detailed explanations about model predictions can be achieved, helping users to better believe in the system and to apply all the recommendations in the perspective of ensuring ethics in AI.

- **Importance of Study**

This will be important to the domain of cybersecurity as it proposes a robust, scalable, and ethical framework of insider threat detection. Temporal data analysis, psychometric profiling, and real-time deployment fill an important gap between theory and practice. The results

confirm not only the effectiveness of the proposed framework but also gave some valuable insights into user behavior which could inform the organization's policies and security measures.

Emphasis on ethical considerations, such as data privacy and explainability, will ensure that the framework will be in line with industry standards and regulatory requirements. Consequently, it enhances the acceptance and adoption of the system in real-world applications. The overall contributions in this research are vital gaps in available methodologies and set the pace for further developments in insider threat detection.

Thus, the proposed framework would serve effectively to highlight deep learning and ensemble modeling for the profiling of organizational threats. The research, therefore, attempts to address the key challenges, incorporating ethical use issues and offering practical solutions, in an effort to respond to the ever-growing demand for cybersecurity advanced tools in dynamic organizational environments. This discussion, therefore, outlines the importance of such findings, underlining the practical implications and offering a direction for future research based on this premise.

5.3. Endpoint Security Analysis

5.3.1. Research Findings

- **Model Performance**

The evaluation of the models focused on their operational classification speed and computational expenditure together with predictive power. The Random Forest model demonstrated outstanding performance by delivering excellent accuracy levels along with high precision and recall scores. The model performed effectively at generalization while maintaining interpretability through its feature importance scores because of its ensemble approach. The system produced fast predictions right away, which made it suitable for Security Operations Center (SOC) environments that require rapid responses. The Neural Network (Multi-Layer Perceptron) model performed more effectively at spotting complex non-linear patterns together with borderline threat scenarios despite attaining inferior total accuracy results. This model processed input slower than other counterparts, yet its ability to handle complex situations was better. The simplified feature framework used within both models

enhanced their response speed and prevented overfitting. The deployed threat detection system used the models to deliver dependable results across distinct traffic simulations and performed effectively alone or combined with additional system components. The hybrid model shows excellence in balancing speed with interpretability and adaptability, which makes it well-suited for endpoint threat detection system deployment.

- **Evaluation Metrics**

The Random Forest model containing reduced features reached an outstanding performance level by recording 99.66% accuracy on the test data. The model achieved full accuracy (1.00) during both classification categories and maintained 0.99 anomaly detection accuracy as well as perfect accuracy (1.00) for standard traffic while registering minimal errors within its results. The exceptional model performance demonstrates strong reliability and its ability to generate interpretable information with decision paths and feature important scores. The Neural Network demonstrated slightly lower performance levels when evaluating security traffic with a total accuracy of 97.0% and precision scores at 0.97 for anomalies and 0.96 for normal traffic. The model recorded satisfactory generalization capabilities with recall rates at 0.96 and 0.97 that proved weaker than RF. The NN needed numerous computational resources to complete multiple training epochs through its multilayer framework when utilizing reduced features. These results reinforce Random Forest as the superior option since it provides higher accuracy levels together with more efficiency in this application domain.

Class	Random Forest Model	Neural Network Model
Precision	1.00 (Anomaly) / 1.00 (Normal)	0.97 (Anomaly) / 0.96 (Normal)
Recall	0.99 (Anomaly) / 1.00 (Normal)	0.96 (Anomaly) / 0.97 (Normal)
F1-Score	1.00 (Anomaly) / 1.00 (Normal)	0.96 (Anomaly) / 0.97 (Normal)
Support	2365 (Anomaly) / 2674 (Normal)	2365 (Anomaly) / 2674 (Normal)
Accuracy	1.00	0.97
Macro avg	1.00	0.97
Weighted avg	1.00	0.97

Figure 17 - Model Accuracy Comparison

5.3.2. Discussion

- Challenges and limitations

Despite the system being effective, some challenges and limitations were encountered in the process of developing the system. The biggest challenge was achieving maximum performance in the Neural Network model. It required lots of hyperparameter tuning, i.e., learning rate, batch size, and number of epochs, making training longer. Additionally, the dataset, though comprehensive, was missing some modern or encrypted forms of attacks, which lessened the model's sensitivity to new attacks. The real-time inference performance of the Neural Network model also had a limitation, particularly on low-spec machines, where latency became more noticeable. Further, the system presently supports only binary classification—normal or anomaly—and not the classification of types of attacks. The use of labeled datasets for supervised learning also puts a constraint on the system to adapt to zero-day attacks. These constraints indicate avenues for future research and development, particularly in the areas of dataset diversity, model flexibility, and computational optimization.

- Future Directions

Based on the existing architecture, there are a number of possibilities for future enhancements. Firstly, the system can be developed to facilitate multi-class classification so that it would be able to identify and classify individual attack types like DoS, probe, or R2L attacks. Secondly, incorporating unsupervised or semi-supervised learning mechanisms can help in identifying zero-day threats and unknown anomalies without depending on labeled datasets. Third, the system can be extended to distributed or cloud environments for performance and availability enhancement for enterprise deployment at scale. Support for encrypted traffic analysis is another crucial enhancement that would enhance the component's applicability in modern networks. Finally, the incorporation of federated learning can potentially allow the model to train on multiple endpoints without data privacy loss, aligning with future trends in decentralized cybersecurity. These future directions are designed to promote accuracy, flexibility, and adherence, rendering the system more applicable in dynamic and hazardous environments.

- Importance of Study

This study is a significant step towards endpoint cybersecurity since it demonstrates that a hybrid machine learning platform is able to effectively detect threats in real-time. The

combination of Random Forest and Neural Network models offers the best balance of accuracy, adaptability, and interpretability—three pillars symbolize the very foundation of modern cybersecurity solutions. The real-time capability of the system, developed using Flask and SocketIO, allows for round-the-clock monitoring and real-time threat identification, an absolute necessity for SOC's. Further, the application of Scapy for simulating traffic offers a controlled environment with realistic conditions, filling the gap between experimentation in academia and actual deployment. The lightweight and modular architecture renders the solution deployable on a wide range of systems, including those with constrained computational resources. Overcoming some of the critical limitations of traditional IDS, such as static data dependency and lack of real-time support, this work offers a pragmatic, scalable, and efficient solution for modern threat scenarios.

5.4. Physical Security Analysis

The system was designed to identify violent activity in surveillance footage with deep learning, emphasizing real-time responsiveness and high precision. Through training the model on the Real-Life Violence Situations Dataset, we successfully classified sequences depicting physical altercations. The aim was not just to detect violence but to achieve this with a minimal false alarm rate, hence ensuring the system's reliability in actual public security contexts.

Visual performance assessments, comprising loss and accuracy graphs, a confusion matrix, and a comprehensive classification report, substantiate the model's resilience and preparedness for real-world implementation.

5.4.1. Research Findings

The training and validation curves demonstrate that the model achieved effective convergence, exhibiting a gradual reduction in loss and a constant enhancement in accuracy throughout 28 epochs. The loss of validation first varied but then stabilized, indicating that the model effectively generalized beyond the training data without substantial overfitting.

The classification report and confusion matrix demonstrate the model's efficacy in differentiating between violent and non-violent scenarios. The algorithm accurately predicted 96 of 101 violent occurrences and 89 of 99 non-violent events, demonstrating robust overall detection capabilities.

○ Model Performance and Evaluation Metrics

The model's performance metrics are summarized as follows:

- **Precision:** 0.91 (violence), 0.95 (non-violence)
- **Recall:** 0.95 (violence), 0.90 (non-violence)
- **F1-Score:** 0.93 (violence), 0.92 (non-violence)
- **Overall Accuracy:** 93%

The **confusion matrix** highlights:

- True Positives (Violence correctly detected): 96
- True Negatives (Non-violence correctly ignored): 89
- False Positives: 10
- False Negatives: 5

These values indicate a well-calibrated model that has a low error rate while demonstrating great sensitivity to aggressive behavior. The macro average F1-score of 0.92 and the weighted average of 0.93 demonstrate constancy among class disparities.

5.4.2. Discussion

The findings confirm that a lightweight CNN model like MobileNet may achieve good accuracy while ensuring rapid inference times. Despite being trained on a fairly large dataset, the model attained performance close to state-of-the-art levels. The minimal false negative rate is especially critical in security systems, as undiscovered aggression may result in severe repercussions.

Nonetheless, false positives albeit infrequent persist, mostly because of unclear frames that mimic violent conduct (e.g., sports or crowd motions). Validation loss variations between epochs 10 and 20 demonstrate sensitivity to illumination and camera discrepancies, a common characteristic of real-world datasets.

○ Challenges and Limitations

Despite promising performance, several challenges emerged:

- **Validation instability** in early training suggests sensitivity to small variations in scene context.

- **Dataset limitations** in representing extreme crowd density or low-light scenarios may restrict performance in complex real-world environments.
- **Temporal ambiguity:** The system processes frame independently, meaning it may miss context available in temporal sequences.

○ Future Directions

To improve system robustness and extend its capabilities, future enhancements may include:

- **Integrating LSTM or 3D CNNs** for capturing temporal behavior in videos.
- **Expanding the dataset** to include more diverse and edge case scenarios.
- **Hybrid audio-visual threat analysis**, using microphones to detect aggression cues.
- **Reducing false alarms** via attention mechanisms or context-aware decision layers.
- **Edge deployment trials** on Jetson Nano and Raspberry Pi to benchmark real-time latency and power usage.

○ Importance of the Study

This research illustrates that deep learning can proficiently automate the identification of violent behavior in surveillance systems with considerable accuracy and little latency. The capacity to detect real-time threats with little human intervention is essential for enhancing public safety. This project establishes a basis for more intelligent, rapid, and scalable surveillance systems by connecting AI research with security applications.

6. COMMERCIALIZATION

The suggested real-time threat profiling system, based on deep learning techniques and tailored for Next-Generation Security Operations Centers (SOCs), has high commercialization potential. This technology surpasses standard security technologies by combining advanced threat detection capabilities from four important domains: network traffic, endpoint behavior, human activity, and physical security into a single, unified solution. Six important strategic pillars contribute to the system's commercial viability:

1. Market Demand

The global cybersecurity market is rapidly rising, owing to the increasing sophistication of cyberattacks, the growing number of connected devices, and the accelerated use of digital technologies. Organizations across industries, from finance and healthcare to government and critical infrastructure, require intelligent, real-time monitoring solutions to protect against advanced threats like zero-day exploits, insider threats, and physical breaches. According to industry statistics, the global SOC-as-a-Service market is quickly growing, providing excellent potential for commercial entry. This technology immediately meets that requirement by offering a scalable, AI-powered solution for proactive threat identification and response.

2. Competitive Advantage

The suggested solution achieves a distinct competitive advantage by merging deep learning models (GANs, CNNs, Neural Networks, and Random Forests) with real-time monitoring tools like Scapy and Socket.IO. Unlike traditional SOC solutions, which depend on rule-based detection or operate in silos, this system integrates different security layers—digital and physical—into a unified analytical framework. The integration of behavioral analytics and physical surveillance sets it apart from most commercial options. Its real-time dashboard, modular design, and advanced data processing techniques significantly improve threat visibility, reduce false positives, and shorten response times.

3. Scalability

The architecture is designed for flexibility and scalability, making it suitable for use in companies of any size. Whether a tiny startup needs endpoint monitoring, or a large business needs full-spectrum threat intelligence across hundreds of devices and sensors, the system may

be quickly scaled to meet unique needs. Cloud compatibility provides horizontal scaling, and the modular component design allows each component (network, human behavior, endpoint, or physical security) to be deployed and maintained independently. Additional modules or data sources can be incorporated with little reconfiguration as the demands of the company increase.

4. Integration Capability

One of this system's primary merits is its ability to seamlessly connect with current security infrastructures. The system is built with widely adopted open-source technologies such as Flask, TensorFlow, Scikit-learn, Keras, and Npcap, and it supports REST APIs and a plugin-based architecture for easy integration with Security Information and Event Management (SIEM) tools, Intrusion Detection Systems (IDS), access control systems, and video surveillance platforms. This makes the solution very customizable and lowers the barrier to implementation for businesses with existing security ecosystems.

5. Subscription-Based Model

To optimize adoption and accessibility, the system is best suited for commercialization via a Subscription-as-a-Service (SaaS) model. Organizations can choose plans based on their infrastructure size, security requirements, and desired capabilities, which range from simple monitoring to full-scale real-time analytics and predictive threat modeling. Subscription tiers may include:

- Basic Plan: Network and endpoint monitoring for small businesses.
- Standard Plan: Includes behavioral profiling and real-time dashboard access.
- Enterprise Plan: Full integration with physical security systems, customization, and 24/7 support.

This pricing flexibility allows businesses to adopt the technology without large upfront investments and supports recurring revenue for the provider.

6. Regulatory Compliance

With firms under increasing pressure to comply with requirements such as GDPR, HIPAA, NIST, and ISO/IEC 27001, this solution provides built-in compliance support. It keeps complete audit trails, logs incidents in real time, and generates comprehensive reports to help with security audits. Furthermore, the behavioral and physical monitoring components provide an extra layer of security by addressing insider threats and workplace safety, both of which are

critical considerations in compliance frameworks. By assisting firms in proactively detecting, documenting, and mitigating risks, the system becomes an advantage for regulatory compliance.

7. BUDGET ALLOCATION

Item	Description	Updated Estimated Cost (LKR)
Development	Software tools, IDE licenses, and development resources	8000
Data Charges	Costs for cloud storage and data processing	12000
Data Collection	Expenses related to data sourcing and preparation	1500
Documentation	Preparation of reports, manuals, and version control	3500
Testing	Model validation, user testing, and performance analysis	2000
Web Application Development	Development of the web app and integration	600
Backend Development	Server-side implementation using Flask-SocketIO	800
Security Measures	Implementation of data security protocols	1000
Miscellaneous Costs	Additional unforeseen expenses and contingency	4000
Total		33,400

Table 9 - Budget Allocation

8. GANTT CHART

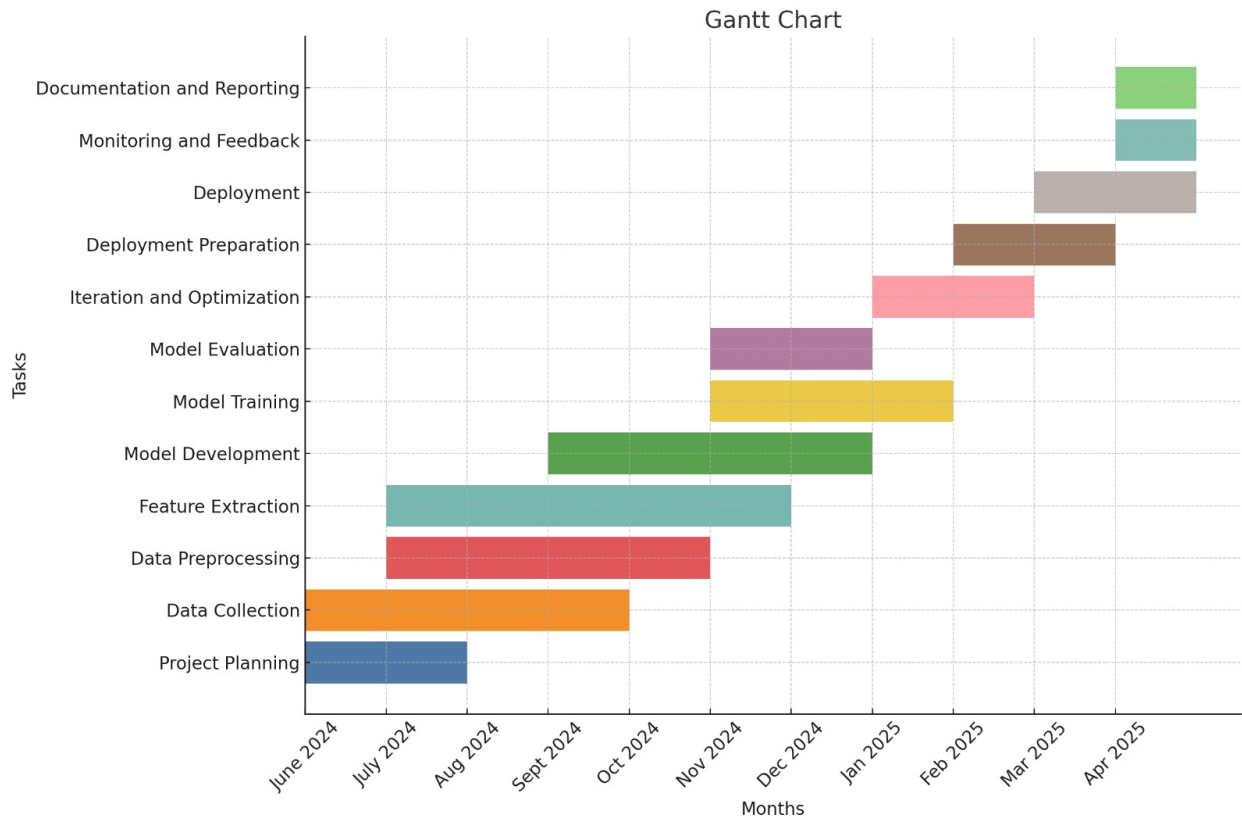


Figure 18 - Gantt Chart

9. CONTRIBUTION

IT Number	Name	Component	Key Contributions
IT21226496	W.M.M. Gunasekara	Network Traffic Analysis	<ul style="list-style-type: none"> - Implemented real-time packet sniffing using Scapy and Npcap - Applied GAN and FNN models for anomaly detection by using CICIDS 2017 dataset - Built backend for traffic flow analysis - Developed web dashboard integration for live threat updates
IT21380396	M.F.F. Ashra	Endpoint Threat Detection	<ul style="list-style-type: none"> - Collected and preprocessed system log and traffic data from endpoint devices - Trained Random Forest and Neural Network models for anomaly detection - Designed real-time endpoint monitoring interface - Implemented Socket.IO-based dynamic communication
IT21298608	D.H. Senevirathna	Human Behavior Profiling	<ul style="list-style-type: none"> - Analyzed behavioral datasets including logon patterns and personality traits (OCEAN model) - Built CNN, Random Forest, and ensemble models for insider threat detection - Designed frontend for user input and real-time profiling - Evaluated model accuracy and performance
IT21058196	K.P.A.T. Gunawardhana	Physical Security Threat Detection	<ul style="list-style-type: none"> - Processed CCTV video feeds using the Real-Life Violence Situations Dataset - Applied MobileNetV2 and LSTM for violence detection - Developed Flask-based video stream handling - Integrated physical threat alerts into the dashboard

Table 10 - Contribution of each Student

10. CONCLUSION

In a day where cyber threats are becoming more frequent, complicated, and devastating, the need for smart, intelligent, and adaptable cybersecurity solutions is greater than ever. This study aimed to overcome the constraints of traditional Security Operations Centers (SOCs) by developing and implementing an integrated, real-time threat profiling system that uses innovative deep learning technology to monitor and detect threats across different vectors. The suggested system presents a transformative approach to cybersecurity operations by combining network traffic analysis, endpoint monitoring, human behavioral analytics, and physical security surveillance into a single, unified platform.

The system architecture was meticulously designed utilizing the Software Development Life Cycle (SDLC) paradigm, which ensured an organized, iterative process from planning and analysis to implementation and continuous maintenance. Each phase helped to refine system requirements, increase design efficiency, and enabled the seamless integration of numerous data processing components. The adoption of SDLC also guaranteed that testing and evaluation were included on every level, allowing for continual validation of performance, functionality, and adaptability.

From a technical standpoint, the research has made considerable progress by employing a variety of deep learning models suited to various sorts of threat data. GANs and Feedforward Neural Networks (FNNs) were trained on the CICIDS2017 dataset to detect anomalies in real time. Endpoint threat detection used Random Forest and Neural Network models to examine system logs and device behavior for indications of compromise. Human behavior profiling, an often overlooked but critical aspect of threat detection, used a combination of CNNs and ensemble approaches to identify insider risks based on personality attributes, login behaviors, and device usage. For physical security, MobileNetV2 and LSTM models were employed to process video feeds, allowing the system to detect aggressive or suspicious activity in real time.

Each of these components was created and tested separately before being combined into a real-time web-based dashboard with Flask and Socket.IO. The dashboard is the primary interface for SOC analysts, offering dynamic visualizations, live warnings, and interactive threat summaries. This combination of modular backend intelligence and responsive frontend enables proactive decision-making while dramatically reducing response time during security issues.

Commercially, the system has enormous potential. Depending on the needs of the organization, it can be deployed as an on-premises solution, a cloud-based SaaS model, or supplied via hybrid architecture. Several critical qualities further assist the commercialization plan, including high market demand, a distinct competitive advantage, scalable infrastructure, seamless integration capabilities, regulatory compliance preparedness, and flexibility via a subscription-based model. These characteristics distinguish the system as not only original research output, but also a feasible solution for enterprise adoption in industries such as banking, healthcare, government, and defense.

Furthermore, this study presents an important academic addition by illustrating the practical application of deep learning in a multi-source, real-time cybersecurity setting. It fills the gap between theoretical AI research and real security implementations by presenting detailed methodology, experimental setting, and performance metrics that can be used as a foundation for future study. The incorporation of numerous datasets—including CICIDS2017, CERT Insider Threat Dataset, and Real-Life Violence Situations Dataset—increases the system's robustness and flexibility to different threat landscapes.

To summarize, this study effectively achieved its goal of establishing an intelligent, comprehensive threat profiling system for Next-Gen SOC's. It provides a comprehensive and future-proof solution that addresses the fundamental difficulties of modern cybersecurity, including fragmented monitoring, delayed reaction, and a lack of contextual intelligence. The system's architecture, performance, and adaptability demonstrate its significance as both an academic milestone and a commercially viable cybersecurity product. Future additions may include integration with automated incident response systems, the deployment of cloud-native microservices, support for new data streams such as dark web monitoring, and extension into mobile and IoT security analytics. Finally, this study provides the framework for the next generation of intelligent cybersecurity infrastructure: smarter, quicker, and more secure.

11. REFERENCES

- [1] M. Vielberth, F. Böhm, I. Fichtinger and G. Pernul, "Security Operations Center: A Systematic Study and Open Challenges," 2020.
- [2] U. N. A. Perera, S. Rathnayaka, N. D. Perera and W. Madushanka, "The Next Gen Security Operation Center," 2021.
- [3] N. Baker, "Advanced Machine Learning Models for Adaptive," 2024.
- [4] A. M. B. M. I. A. F. E. S. H. B. S. A.-N. & S. S. A.-N. Msbah J. Mosa, "AI and Ethics in Surveillance: Balancing Security and Privacy in a Digital World," 2024.
- [5] M. Vielberth, F. Böhm, I. Fichtinger and G. Pernul, "Security Operations Center: A Systematic Study and Open Challenges," 2020.
- [6] S. Schinagl, K. Schoon and R. Paans, "A Framework for Designing a Security Operations Centre (SOC)," 2015.
- [7] Sateeshkumar6289, "'CICIDS 2017 Dataset,'" Kaggle.com, 2017
<https://www.kaggle.com/datasets/sateeshkumar6289/cicids-2017-dataset>.
- [8] H. e. al., "Evolution of Security Operation Centers and Their Role in Cybersecurity," 2019.
- [9] I. Security, "What is a Security Operations Center (SOC)?," 2021.
- [10] "What is a Security Operations Center (SOC)?," paloalto, [Online]. Available:
<https://www.paloaltonetworks.com/cyberpedia/what-is-a-soc>.
- [11] Y. L. Z. J. M. F. G. & S. V. Yin, "Practical GAN-based synthetic IP header trace generation using NetShare," 2022.
- [12] N. L. Z. K. D. B. G. B. & R. K. Thapa, "Comparison of Machine Learning and Deep Learning Models for Network Intrusion Detection Systems," Future Internet, 12(10), 167., 2020.
- [13] H. A. R. T. C. C. J.-N. B. E. & B. X. Hindy, "Utilising Deep Learning Techniques for Effective Zero-Day Attack Detection," 2020.
- [14] R. A. M. S. K. P. P. P. A.-N. A. & V. S. Vinayakumar, "Deep Learning Approach for Intelligent Intrusion Detection System.," IEEE Access, 7, 41525–41550, 2019.

- [15] H. A. H. A. & B. N. Z. hmed, "Network intrusion detection using oversampling technique and machine learning algorithms," *PeerJ Computer Science*, 8, e820., 2022.
- [16] Y. B. M. M. Y. M. Y. S. A. B. D. & E. Y. Meidan, "N-BaloT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*,, 2018.
- [17] Y. T. Junkai Yi, "Insider Threat Detection Model Enhancement Using Hybrid Algorithms between Unsupervised and Supervised Learning," 2024.
- [18] O. S. A. B. K. A. O. A. Olajide O. Ogunbodede, "Insider Threat Detection Techniques: Review of User Behavior Analytics Approach," 2024.
- [19] L. K. V. G. S. R. D. Sridevi, "Detecting Insider Threats in Cybersecurity Using Machine Learning and Deep Learning Techniques," 2023.
- [20] K. S. D. G. J. H. M. S. a. R. R. Gaurang Gavai, "Supervised and Unsupervised methods to detect Insider Threat from Enterprise Social and Online Activity Data," 2015.
- [21] Z. A.-I. Y. T. C. F. Ahmed hasan Saaudi, "Insider Threats Detection using CNN-LSTM Model," 2019.
- [22] H. J. H. A. B. C. Shao Hong Zhong, "An Effective Intrusion Detection Model Based on Random Forest and Neural Networks," *reserchGate*.
- [23] a. Y. M. Wenjuan Li, "Improving the Performance of Neural Networks with Random Forest in Detecting Network Intrusions," *Springer*, 2013.
- [24] S. A. C. G. D. E. Brian Lee, "comparative Study of Deep Learning Models for Network," 2018.
- [25] P. D. P. Roßbach, "Neural Networks vs. Random Forests," 2018. [Online]. Available: <https://blog.frankfurt-school.de/wp-content/uploads/2018/10/Neural-Networks-vs-Random-Forests.pdf>.
- [26] B. I. Mohammed F. Suleiman, "Performance Comparison of Intrusion Detection," https://researchportal.northumbria.ac.uk/ws/portalfiles/portal/17488130/Accepted_manuscript_ICCTA_2018.pdf.

- [27] S. W. Raisa Abedin Disha, "Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique," *springeropen*, 2022.
- [28] J. M. U. A. Amos Oyetoro, "Using Machine learning Techniques Random Forest and Neural Network," *Scienceopen*, 2023.
- [29] C. I. T. D. Research. [Online]. Available: <https://www.kaggle.com/datasets/mrajxnp/cert-insider-threat-detection-research>.
- [30] Y. L. Z. J. M. F. G. & S. Yin, "Practical GAN-based synthetic IP header trace generation using NetShare," 2022.
- [31] N. L. Z. K. D. B. G. B. & R. K. Thapa, "Comparison of Machine Learning and Deep Learning Models for Network Intrusion Detection Systems," 2020.
- [32] P. L. P. L. Q. L. C. L. X. H. R. & C. J. Sun, "DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System," 2020.
- [33] R. A. M. S. K. P. P. P. A.-N. A. & V. S. Vinayakumar, "Deep Learning Approach for Intelligent Intrusion Detection System," IEEE, 2019.
- [34] H. A. H. A. & B. N. Z. Ahmed, "Network intrusion detection using oversampling technique and machine learning algorithms," 2022.
- [35] Y. B. M. M. Y. M. Y. S. A. B. D. & E. Y. Meidan, "Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders," IEE, 2022.
- [36] L. S. S. B. Anagi Gamachchi, "A Graph Based Framework for Malicious Insider Threat Detection," 2018.
- [37] O. S. A. B. K. A. O. A. Olajide O. Ogunbodede, "Insider Threat Detection Techniques: Review of User Behavior Analytics Approach," 2024.
- [38] L. K. V. G. S. R. D. Sridevi, "Detecting Insider Threats in Cybersecurity Using Machine Learning and Deep Learning Techniques," 2023.
- [39] S. S. M. F. M. R. K. M. R. I. Y. W. Md Abrar Jahin, "CAGN-GAT Fusion: A Hybrid Contrastive Attentive Graph Neural Network for Network Intrusion Detection," 2025.
- [40] Z. A.-I. Y. T. C. F. hmed hasan Saaudi, "Insider Threats Detection using CNN-LSTM Model," 2019.

- [41] V. S. B. R. E.-E. Y. A. B. T. P. S. K. A. & G. S. R. Rao, "AI Driven Anomaly Detection in Network Traffic Using Hybrid CNN-GAN.," *Journal of Advances in Information Technology*, 15(7), 886–895..
- [42] P. L. P. L. Q. L. C. L. X. H. R. & C. J. Sun, "DL-IDS: Extracting Features Using CNN-LSTM Hybrid Network for Intrusion Detection System," *Security and Communication Networks*, 2020, 1–11., 2020.

12. APPENDICES

1. TAF



RETAF_24-25J-075.pdf
f

2. Plagiarism Report