Research Project Proposal

Compound Prediction Analysis of Information Network Security Situation Based on Support Vector Combined with Classification Algorithm

Group: 18

Danujan S.	E/19/060
Harnan M.	E/19/134.
Nithusikan T.	E/19/266

Department of Computer Engineering
Faculty of Engineering
University of Peradeniya
2025

ABSTRACT

The increasing sophistication of cyber threats targeting web applications poses significant challenges for traditional security mechanisms. Identifying the vulnerable components of a web application during a cyber attack is crucial for mitigating security risks and enhancing system resilience. Attackers exploit weaknesses in web applications and APIs, often bypassing conventional security defenses.

In this work, we propose a framework to detect and analyze vulnerable components by correlating system logs and access logs with identified cyber attacks. First, network traffic is classified to distinguish between normal requests and attack patterns. If an attack is detected, the corresponding application logs are retrieved based on the time interval of the attack. By analyzing these logs, we determine which application function, service, or module is susceptible to exploitation.

The framework employs a three-stage classification approach. Initially, supervised classifiers are used to identify known attack patterns. Next, an anomaly detection model, specifically a One-Class SVM, classifies previously unseen or anomalous traffic. Finally, a root cause analysis component processes application logs within the detected attack timeframe to accurately determine the vulnerable component.

Designed for seamless integration into existing security infrastructures, such as web application firewalls (WAFs) and API gateways, the proposed framework operates efficiently without disrupting normal workflows. Performance evaluation is conducted based on detection accuracy, computational efficiency, and adaptability to evolving cyber threats. This approach enhances application-layer security by providing actionable insights into vulnerable components, aiding in proactive threat mitigation during software development and deployment.

Keywords: Cyber Attacks, Web Application Firewalls, Compound Prediction, Neural Networks, Network Security Situation.

Contents

List of Figures

CHAPTER 1

INTRODUCTION

1.1 Overview

1.1.1 Background

The rapid digital transformation has led to a significant increase in cyber threats, particularly at the application layer, where web applications and microservices are prime targets for attackers. Traditional Intrusion Detection Systems (IDS) and Web Application Firewalls (WAFs) primarily rely on rule-based or signature-based mechanisms to detect malicious activities. However, these approaches struggle against modern cyber threats, such as zero-day attacks and sophisticated obfuscation techniques, which bypass conventional security measures [19]. Signature-based detection methods require constant updates to maintain effectiveness, yet they remain vulnerable to novel attacks that do not match predefined patterns [2],[7].

Machine learning (ML) and deep learning (DL) have emerged as powerful tools in the field of cybersecurity, offering the ability to detect both known and unknown threats by learning from historical attack patterns [14],[16]. Research has demonstrated that ML-based IDS solutions can enhance threat detection accuracy, reduce false positives, and adapt to new attack patterns more efficiently than rule-based approaches [24],[13]. However, existing ML-based IDS solutions primarily focus on attack classification rather than identifying the root cause of vulnerabilities within an application [7],[18]. Without a deeper forensic analysis of system behaviors, current IDS solutions fail to provide security teams with precise insights into which web application function or service is being exploited [6],[22].

To address these challenges, this study proposes a hierarchical intrusion detection framework that integrates supervised classification techniques with anomaly detection while incorporating forensic analysis of application logs to identify the specific affected function or service. By correlating network traffic, system logs, and application-layer behaviors, this approach enhances the precision of intrusion detection and facilitates actionable mitigation strategies [6],[22].

1.1.2 Importance of Intrusion Detection at the Application Layer

The increasing reliance on web-based services has made application-layer attacks, such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Remote Code Execution (RCE), more prevalent [21],[7],[18]. Unlike network-layer attacks, which primarily target packet transmissions and connection states, application-layer attacks exploit vulnerabilities within the logical structure of web applications, allowing adversaries to manipulate requests, inject malicious code, and gain unauthorized access to sensitive data [5],[24]. Traditional IDS and WAF solutions often fail to detect such attacks, as they rely on static rules that cannot adapt to evolving evasion techniques used by cybercriminals [6],[13].

Modern IDS solutions must move beyond static, rule-based detection methods by integrating intelligent, adaptable methodologies that can detect emerging threats in real-time. ML and DL-based security mechanisms improve intrusion detection by reducing false positives and adapting to new attack vectors dynamically [19],[13]. However, many existing approaches lack an integrated strategy for correlating attack indicators across system logs, application logs, and network activity. Without such correlation, it is difficult to trace an attack back to the specific vulnerable component within an application [8],[4]. Identifying these weaknesses is crucial for mitigating risks effectively and preventing future exploits.

1.1.3 Research Motivation

The motivation for this research stems from the need for an advanced intrusion detection approach that addresses the limitations of current IDS solutions. The proposed framework integrates multiple detection techniques into a structured, hierarchical architecture, enhancing scalability, accuracy, and adaptability to diverse and evolving cyber threats [14],[27]. The inclusion of a hybrid detection approach allows for the classification of known attack types using supervised learning, while an anomaly detection model ensures the identification of previously unseen attack patterns [24],[6].

By leveraging application-layer logs, the system extracts meaningful features that improve the detection of subtle attack behaviors often missed by traditional IDS models [18],[7].

Furthermore, the integration of neural networks and ensemble-based machine learning techniques optimizes the balance between detection accuracy and computational efficiency, making the solution viable for resource-constrained environments [16],[27]. The proposed framework also includes a forensic analysis component that correlates system, application, and access logs to provide a comprehensive root cause analysis of security incidents. This enhances threat response efficiency by pinpointing the specific service or function affected by an attack, thereby facilitating rapid and targeted mitigation [6],[8].

By implementing these innovations, the framework not only enhances intrusion detection capabilities but also provides security analysts with actionable insights for improving the resilience of modern web applications. Unlike traditional IDS solutions that focus solely on attack identification, this framework aims to bridge the gap between attack detection and vulnerability analysis, ensuring that security teams can address the underlying causes of cyber threats effectively [19],[23].

1.1.4 Challenges in Existing IDS Solutions

Despite advancements in intrusion detection systems, several challenges hinder their effectiveness. One of the most significant challenges is the high false positive rate, where legitimate activities are mistakenly flagged as malicious, leading to unnecessary security alerts and resource exhaustion [13],[6]. Many anomaly detection techniques suffer from this issue, as they often classify benign deviations from normal traffic patterns as potential threats. This reduces the operational efficiency of security teams and increases response times for real incidents.

Another critical challenge is the emergence of adversarial attacks, where attackers craft inputs specifically designed to deceive machine learning models into misclassifying malicious activities as benign [1],[23]. These adversarial threats pose a significant risk to ML-based IDS solutions, as small perturbations in input data can drastically alter model predictions. Current IDS frameworks lack effective adversarial defense mechanisms, leaving them susceptible to evasion techniques employed by sophisticated attackers [16],[9].

The lack of generalization in existing IDS models further complicates intrusion detection. Many IDS solutions are trained on outdated or static datasets, limiting their ability to recognize new and evolving attack patterns [14],[9]. The dynamic nature of cyber threats requires IDS models to continuously adapt and retrain on updated datasets. However, many existing systems fail to incorporate mechanisms for ongoing learning, making them less effective in real-world scenarios [27].

The complexity of application-layer attacks introduces additional difficulties, as these threats often involve a combination of structured and unstructured data that require advanced feature extraction techniques [7],[30]. Unlike network-layer attacks, which primarily involve packet-level analysis, application-layer threats manipulate textual inputs, HTTP requests, and session states, making them harder to detect without deep semantic analysis. Many IDS solutions lack the capability to process such diverse data formats, reducing their effectiveness in identifying sophisticated web-based exploits [18],[13].

Finally, most IDS solutions fail to establish cross-layer correlations between security events, analyzing network traffic, application logs, and system logs in isolation. This fragmented approach prevents security analysts from identifying the exact software components affected by an attack, prolonging incident response times and increasing the likelihood of repeated exploits [6],[8],[4]. Without a comprehensive correlation framework, organizations struggle to trace attacks back to their origin, leading to incomplete threat mitigation strategies and increased security risks.

Addressing these challenges requires a novel approach that integrates hierarchical detection mechanisms, cross-layer data correlation, and automated vulnerability tracing. The proposed framework provides a robust, scalable, and efficient solution for mitigating modern cybersecurity threats by bridging the gap between intrusion detection and vulnerability analysis [19],[23],[27]. By pinpointing the specific vulnerable component within an application, this framework enhances response times, facilitates automated remediation, and strengthens the overall security posture of modern web-based systems [24],[18].

1.2 PROBLEM STATEMENT

Modern cyber threats increasingly target the application layer of web infrastructures, exploiting vulnerabilities in web applications and APIs while bypassing traditional security measures such as rule-based firewalls and signature-based intrusion detection systems. These threats evolve rapidly, leveraging obfuscation techniques and zero-day exploits, making conventional static defense mechanisms ineffective. Additionally, the complexity of modern distributed systems introduces significant challenges in distinguishing legitimate user traffic from malicious payloads, often leading to high false-positive rates in intrusion detection.

A major limitation of current security solutions is their inability to correlate attack indicators across multiple layers, including system logs, application logs, and network activity. Traditional intrusion detection approaches often analyze attacks at a single level, missing critical contextual relationships between anomalies in system behavior and application-layer failures. Without a cross-layer correlation mechanism, security teams struggle to identify the specific software component or service affected by an attack, leading to delayed responses and ineffective mitigation.

The primary objective of this research is to develop a hierarchical, multi-stage intrusion detection framework that not only detects cyberattacks but also identifies the specific vulnerable component of the application during an attack. The proposed approach integrates supervised classification for known attack detection, anomaly detection for novel threats, and root cause analysis through application log correlation. By leveraging system logs for attack detection and application logs for forensic analysis, this framework aims to pinpoint which microservice, function, or module is impacted by an attack. This enhances precision, reduces false alarms, and provides a structured methodology for identifying and mitigating vulnerabilities at the application layer. Furthermore, the proposed approach is designed for seamless integration into existing Web Application Firewalls (WAFs), API gateways, and security monitoring tools, ensuring its applicability in modern security infrastructures.

1.3. AIM AND OBJECTIVES

1.3.1 Aim

To design, implement, and evaluate a hierarchical, multistage detection system capable of identifying both known and unknown web application attacks as well as the application component which is vulnerable during the cyber attacks with high accuracy, scalability, and adaptability, thereby enhancing overall application-layer security.

1.3.2 Objectives

1. To design and develop a hierarchical, multi-stage algorithmic framework
2. To define measurable security metrics, including classification accuracy, detection precision, computational efficiency, and adaptability
3. To explore several algorithmic approaches within the hierarchical framework, combining neural networks with ensemble-based machine learning techniques
4. To design scalable algorithms capable of adapting to future application-layer attack innovations and network expansions
5. To investigate practical techniques for integrating the proposed framework into real-world application-layer security systems

By addressing these objectives, this research aims to contribute significantly to the field of web application security, providing robust, scalable, and efficient solutions to mitigate the growing threat landscape.

CHAPTER 2

Existing Approaches and Challenges in Detecting Cyber Security Attacks

2.1 Introduction to Intrusion Detection Systems and Cybersecurity Threats

Intrusion Detection Systems (IDS) serve as a crucial layer in modern cybersecurity infrastructures by identifying and mitigating malicious activity within networks and web applications before such threats escalate [19]. Historically, IDS relied on signature-based detection [2], where predefined attack patterns were stored in databases and compared against incoming network traffic. Although effective against documented threats, these systems struggle to detect zero-day exploits and often exhibit high false-positive rates when signatures are outdated or incomplete [7],[6]. The cybersecurity community has therefore turned to machine learning (ML) and deep learning (DL) to address these shortcomings, leveraging their ability to learn from historical attack patterns and detect anomalies in real-time[1],[24].

Early ML-based IDS implementations, such as Support Vector Machines (SVM) [10],[11],[12] Decision Trees, and Random Forests, significantly reduced false-positive rates while improving detection accuracy [24]. More recent advancements in DL-based IDS, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Transformer-based models, have further improved intrusion detection capabilities by effectively handling large and high-dimensional datasets [4].

2.2 Network-Layer vs. Application-Layer Intrusion Detection

Researchers have noted that IDS technologies must be capable of detecting both network-layer and application-layer threats. Network-layer IDS focuses on packet headers and aggregated network flows to identify threats such as Distributed Denial of Service (DDoS), botnets, and port scanning [10],[3]. Application-layer IDS, on the other hand, detects attacks that exploit web application logic, such as SQL injection (SQLi), cross-site scripting (XSS), and remote file inclusion (RFI) [7],[6],[30],[18].

Although network-layer IDS can efficiently detect volumetric attacks, they struggle to detect application-layer attacks since these are often embedded within legitimate HTTP traffic. This has led to the widespread adoption of Web Application Firewalls (WAFs), such as MODSECURITY, for application-layer defense. However, rule-based WAFs suffer from high false-positive rates, especially when attackers use payload obfuscation techniques [7],[30],[13]. These challenges highlight the need for more adaptive, ML-based approaches that can dynamically identify and mitigate application-layer threats.

2.3 Limitations of Traditional IDS Approaches

A significant drawback of signature-based IDS is its overreliance on static pattern sets, which require frequent updates to remain effective. When an attack is previously unseen or obfuscated, traditional IDS fails to detect it, resulting in high rates of both false positives and false negatives [19],[7],[29]. Compounding this problem is the fact that modern enterprise networks generate vast volumes of data, making offline or slow analysis impractical. Research has shown that even high-throughput distributed IDS can become overwhelmed if not optimized for real-time processing [3],[27].

The limitations of rule-based and static detection mechanisms have led to an increased focus on ML and DL-based IDS. However, these approaches are not without their own challenges. For instance, although supervised ML classifiers like SVM, Decision Trees, Naïve Bayes, and Random Forests have shown strong classification performance, they require well-labeled datasets, which are often difficult to obtain in rapidly evolving cyberattack environments [9],[1].

2.4 Advancements in ML and DL for Intrusion Detection

To improve IDS effectiveness, researchers have explored ML and DL-based detection techniques. Early supervised ML methods, such as Random Forest and SVM, achieved high detection accuracy and low false-positive rates when trained on modern cybersecurity datasets [26],[8]. Various optimization techniques have also been applied to fine-tune these models. For example, Particle Swarm Optimization (PSO) has been used to improve SVM performance, while the Whale Optimization Algorithm (WOA) has been employed to optimize detection models [10],[30].

While supervised ML remains a strong foundation for IDS, unsupervised and semi-supervised approaches have been explored to address dynamic attack patterns. One-Class SVM and clustering-based anomaly detection techniques have shown potential in detecting zero-day threats, but they often suffer from high false-positive rates when legitimate traffic deviates from expected patterns [6],[28]. Hybrid ML models, combining multiple learning techniques, have demonstrated near 99% classification accuracy in controlled experiments using datasets such as NSL-KDD [12]. However, these models are often complex and may not translate well into real-world production traffic [24].

Deep learning architectures have further expanded IDS capabilities. CNN-based IDS has been effective in detecting attack patterns in raw network packets and HTTP logs, while LSTM models have successfully captured temporal dependencies in network activity [19],[28]. Studies integrating CNN and LSTM have demonstrated improved performance on datasets such as CSIC2010 and CICIDS2017, achieving higher accuracy than classical ML models [19],[28]. However, deep learning models pose computational challenges, making them difficult to deploy in real-time enterprise environments [25]. Transformer-based models, which employ self-attention mechanisms, address some limitations of RNNs by handling long-range dependencies in network traffic, but their high computational cost makes them difficult to implement in time-sensitive IDS applications [22],[29].

2.5 Identifying Vulnerable Web Application Components

Although ML and DL-based IDS solutions have improved intrusion detection accuracy, they fail to provide insights into which application components are being exploited during an attack. Existing IDS approaches classify traffic into attack vs. non-attack categories, but they do not identify which web application functions, APIs, or services are vulnerable [7],[6],[18],[13].

Some researchers have proposed enhancing WAFs with ML-based anomaly detection, using textual analysis of HTTP request logs to reveal malicious payloads [6]. However, these solutions rely heavily on labeled datasets, which quickly become outdated due to the rapid evolution of attack strategies [7],[6],[18],[13]. Without real-time adaptability, such models may lose effectiveness in practical security deployments.

2.6 Challenges in IDS Datasets and Evaluation

The reliability of IDS solutions depends on high-quality training datasets. Many widely used benchmark datasets, such as KDD Cup 1999 and DARPA, have been criticized for containing outdated attack scenarios and redundant data [20],[9],[1]. More modern datasets, such as NSL-KDD, UNSW-NB15, and CICIDS2017, have attempted to address these shortcomings by incorporating contemporary attack vectors [3],[18],[14]. However, even these datasets risk becoming obsolete as cyber threats evolve.

For application-layer security, datasets like CSIC2010 are valuable for detecting web-based threats, but the rapid evolution of attack techniques makes it difficult to maintain up-to-date training data [7],[6],[13],[8]. To overcome this, some researchers use synthetic attack data, generated via tools like OWASP Zap and Burp Suite, to create reproducible cyberattack scenarios [22],[13]. However, synthetic datasets lack the complexity of real-world attacks, limiting their effectiveness for real-world IDS evaluations.

2.7 Conclusion and Research Direction

Although ML and DL-based IDS have significantly improved intrusion detection accuracy, several key challenges remain. Scalability, adversarial robustness, and real-time adaptability are critical concerns, especially as attackers develop more sophisticated evasion techniques [1],[23]. ML-based IDS must be robust against adversarial attacks, where malicious inputs are crafted to deceive detection models [1],[23]. Additionally, the integration of ML-based IDS into SIEM (Security Information and Event Management) systems remains a challenge, as organizations must balance detection accuracy, computational cost, and operational feasibility [20],[23].

To address these limitations, this research proposes a framework that correlates network traffic and application logs to identify which web application components are vulnerable during an attack. By combining supervised learning, anomaly detection, and root cause analysis, this approach provides actionable security insights, bridging the gap between intrusion detection and vulnerability analysis.

CHAPTER 3

PROPOSED METHODOLOGY

This section describes the methodology employed to design, implement, and evaluate the proposed hierarchical, multistage detection system for identifying both known and unknown attacks against web applications as well as the component which is vulnerable during the attack. The methodology encompasses the following steps: environment setup, data collection, data preprocessing, model development, hierarchical, multistage detection pipeline integration, and evaluation.

3.1. Overview

Traditional Intrusion Detection Systems (IDS) and Web Application Firewalls (WAF) rely heavily on known attack signatures, making them susceptible to novel threats and also they are not capable of detecting the vulnerable component of the application. To address these limitations, we propose a hierarchical, multistage approach that combines:

1. Supervised Classification for known threats.
2. Anomaly Detection for suspicious, potentially unknown attacks.
3. Root Cause Analysis for identifying the specific vulnerable application component using log correlation.

3.2. Data Collection

The data collection phase involves generating both legitimate (normal) and malicious traffic to create a labeled dataset. Additionally, we collect system logs and application layer logs for forensic analysis.

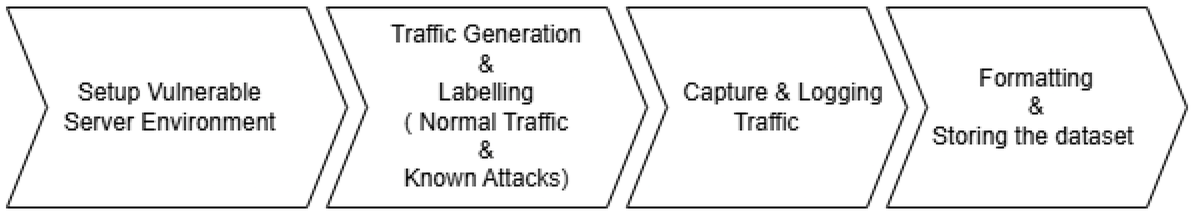
Setup Vulnerable Server Environment
Traffic Generation & Labelling (Normal Traffic & Known Attacks)
Capture & Logging Traffic
Formatting & Storing the dataset

Figure 1: Data collection flow

3.2.1 Setup Vulnerable Server Environment

We deploy intentionally vulnerable web applications like Damn Vulnerable Web Application (DVWA), OWASP Juice Shop, or WebGoat in a controlled setting. These applications are chosen specifically because they contain common vulnerabilities such as SQL Injection (SQLi) and Cross-Site Scripting (XSS). For controlled experimentation, these vulnerable applications are hosted on virtual machines or Docker containers, ensuring each application instance can be monitored independently without affecting other services. Finally, we install logging/monitoring tools or use custom log collection scripts to capture incoming HTTP requests and relevant metadata. System and application logs are collected to establish correlations between detected attacks and the affected application components.

3.2.2. Traffic Generation and Labelling

To generate realistic traffic, we use automation tools like Selenium, JMeter, or Locust to simulate typical user behavior. This includes actions such as visiting pages, filling out forms, uploading files, and logging into user accounts. All automated requests during these normal user simulations are tagged as "normal" in the log repository in a later stage. Following this, we inject attacks using security testing tools like OWASP ZAP or Burp Suite to exploit known vulnerabilities. These tools launch attacks such as SQL Injection, XSS, file inclusion, and other common attack vectors. We also use custom scripts, written in Python or bash, to generate additional malicious payloads and broaden the range of known attack vectors. All malicious requests injected during this attack phase are tagged as "malicious-known" in the log repository.

3.2.3. Log Collection and Correlation

The log collection methodology proposes a structured approach for log collection and correlation to enhance the detection and analysis of cybersecurity threats. The process begins with identifying the time frame of a detected attack by analyzing system logs, which serve as a reference point for extracting relevant data. Once this time frame is established, we systematically collect all application logs within this period to facilitate comprehensive analysis. Web application logs are gathered using security tools such as Burp Suite, OWASP ZAP, SQLMap, and XSS payloads, which aid in identifying vulnerabilities and potential attack vectors. Log retrieval is performed using grep-based commands or centralized logging solutions like ELK Stack (Elasticsearch, Logstash, Kibana), Splunk, or Python scripts, ensuring efficient indexing and searchability.

For database-level monitoring, our approach incorporates MySQL, MongoDB, and PostgreSQL query logs, which provide insights into unauthorized access, SQL injections, and anomalous queries. These logs are aggregated using ELK, Splunk, and Graylog, allowing for real-time visualization and pattern detection. Once the logs are collected, we conduct anomaly correlation analysis to identify patterns that exhibit a high correlation with the detected attack. This process enables us to pinpoint the affected application function or service and facilitates targeted incident response. By integrating a comprehensive log collection framework with anomaly-based correlation analysis, our proposed methodology aims to enhance real-time cybersecurity threat detection and mitigation, improving the overall resilience of applications against evolving attack strategies.

3.3. Data Preprocessing

Following the collection of raw log data, several preprocessing steps are undertaken to prepare the data for model training. This begins with feature extraction, where we derive numerical features such as request size, parameter count, HTTP method, and the presence of suspicious keywords. We also perform log anomaly detection to identify irregular patterns within the application logs during the identified attack time frame. Next, data cleaning is performed, which includes removing redundant entries by filtering out incomplete or duplicated requests.

Anonymization is also carried out to mask sensitive information and ensure compliance with privacy standards. Finally, the dataset is split into training, validation, and test sets to ensure robust model evaluation and prevent overfitting.

3.4. Model Development

3.4.1 Supervised Classifier

For supervised classification, the objective is to distinguish between normal requests and known malicious requests using training data that includes both normal and malicious-known labeled samples, with common algorithms such as Logistic Regression, Random Forest, Gradient Boosting, or Neural Networks, and the output is a binary classification label indicating whether a request is normal or malicious, along with a confidence score.

3.4.2 Anomaly Detector

For anomaly detection, the objective is to identify requests that deviate significantly from normal traffic patterns, potentially indicating unknown or novel attacks, using training data that includes only normal labeled samples to learn the distribution of legitimate requests, with candidate algorithms such as One-Class SVM, Isolation Forest, or Autoencoder-based approaches, and the output is an anomaly score indicating how unusual the request appears, which can be thresholded to flag suspicious requests.

3.4.3 Vulnerability Identification Component:

Vulnerability localization aims to identify which microservice, function, or module is most likely vulnerable during an attack by correlating attack patterns with system and application logs to detect recurring vulnerabilities, and the output is a ranked list of potentially vulnerable components within the application, helping security teams prioritize mitigation efforts.

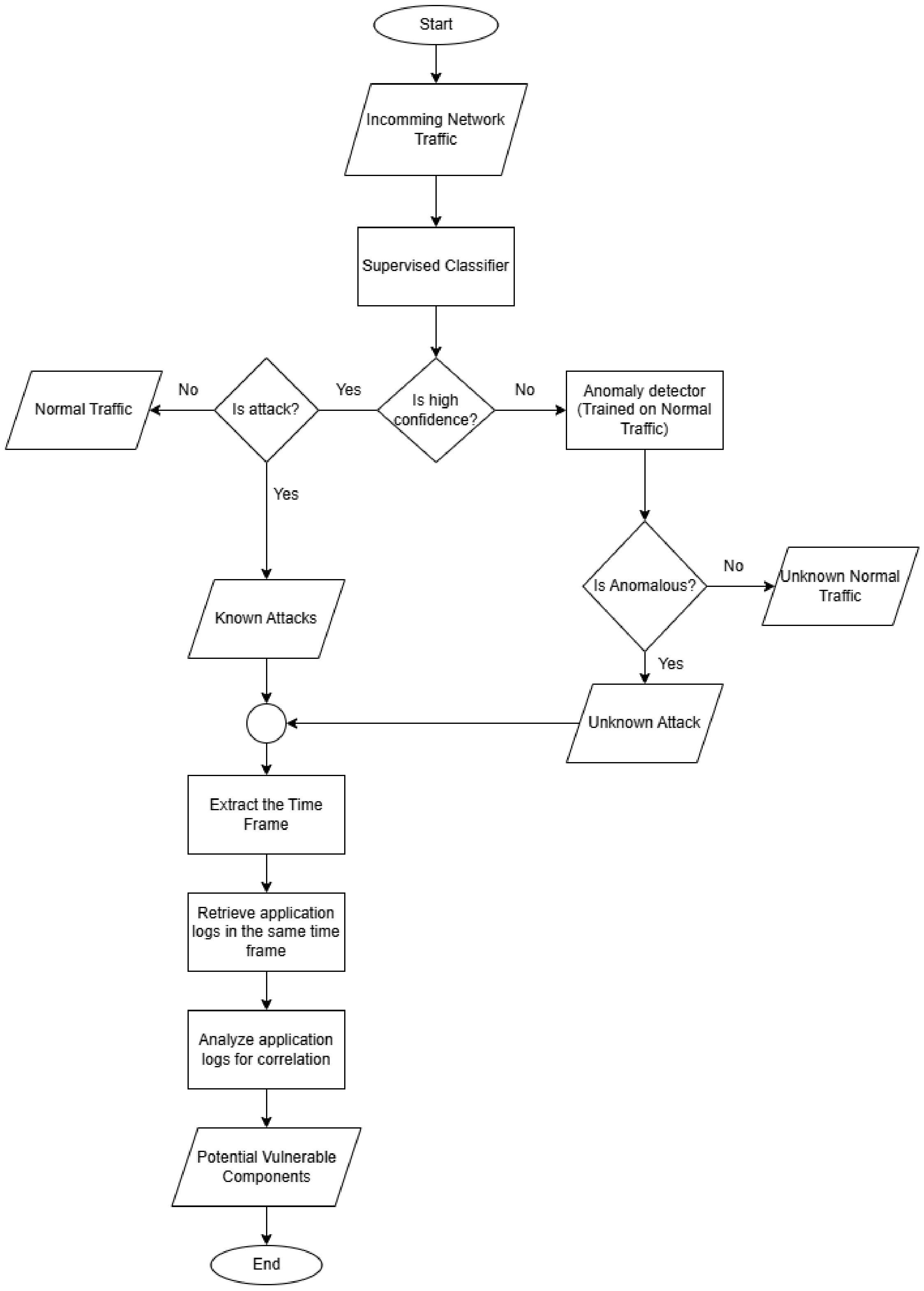
Start
Incomming Network Traffic
Supervised Classifier
Is high confidence?
Yes
No
Is attack?
No
Normal Traffic
Yes
Known Attacks
Anomaly detector (Trained on Normal Traffic)
Is Anomalous?
No
Unknown Normal Traffic
Yes
Unknown Attack
Extract the Time Frame
Retrieve application logs in the same time frame
Analyze application logs for correlation
Potential Vulnerable Components
End

Figure 2: Hybrid pipeline & Component identifier

3.5. Hierarchical, Multistage Detection Pipeline

To combine the strengths of supervised learning and anomaly detection, we propose a two-stage pipeline followed by root cause analysis via log correlation. The process begins with feature extraction , where each incoming request is converted into a feature vector using the same methodology applied during the training phase. Next, the classifier decision stage involves the supervised classifier assigning a label (normal or malicious) to the request, accompanied by a confidence score. If the confidence score exceeds a predefined threshold (Θ), the system directly adopts the classifier's decision, categorizing the request as either a "Known Attack" or "Normal." However, if the confidence score is below the threshold, the request is forwarded to the anomaly detector for further inspection. In this stage, the anomaly detector evaluates whether the request aligns with the learned "normal" data distribution. Requests that deviate significantly are flagged as potential "Unknown Attacks," while those within the normal range are treated as benign, though they may represent an "unknown normal" case. Following detection, root cause analysis is performed through log correlation. This involves retrieving system logs to establish the attack time frame, extracting application logs corresponding to that period, and identifying logs with high correlation to the attack signature. By analyzing these logs, the system determines the most affected microservice, function, or module, enabling security teams to pinpoint vulnerable components. Finally, a feedback loop allows security analysts to review flagged requests, confirming whether they represent genuinely new attacks or benign variations of normal traffic. Confirmed new attack patterns can then be incorporated into future model retraining, ensuring continuous improvement of the system's detection capabilities.

3.6 Evaluation

The performance of the proposed system is evaluated using a variety of metrics and experimental scenarios to ensure comprehensive analysis. Performance metrics include accuracy, which measures the fraction of correctly classified requests over the total, and precision, recall, and F1-score, which serve as key indicators of the system's ability to detect malicious activities effectively. Additionally, the false positive rate (FPR) and false negative rate (FNR) are critical for understanding the trade-off between the cost of false alarms and the risk of undetected attacks. Root cause accuracy evaluates how often the system correctly identifies the affected function or service during an attack, while scalability performance assesses the system's ability to handle large volumes of traffic and logs, ensuring its suitability for real-world deployment.

To test the system's capabilities, several experimental scenarios are designed. The first scenario focuses on known attacks, evaluating how accurately the classifier identifies standard malicious payloads. The second scenario involves unknown or mutated attacks, where slightly altered payloads or new exploit types are submitted to test the anomaly detector's ability to flag unfamiliar threats. A third scenario examines unknown normal cases, assessing the pipeline's tendency to mislabel unusual but legitimate requests as malicious. Additionally, the system's robustness is tested against adaptive attack resistance, measuring its ability to withstand adversarial techniques designed to evade detection.

Threshold tuning plays a crucial role in balancing false positives and false negatives. By varying the classifier confidence threshold (Θ), we analyze the system's performance using ROC (Receiver Operating Characteristic) curves and Precision-Recall curves to guide optimal threshold selection based on different security requirements.

The hybrid system is also compared to two baseline approaches:

(1) a standalone supervised classifier

(2) a standalone anomaly detector.

This comparison highlights the added value of combining both methods, demonstrating the hybrid system's superior performance. Finally, the effectiveness of log correlation is evaluated through root cause analysis verification, which assesses the system's accuracy in pinpointing the vulnerable microservice, function, or module during an attack. Cross-layer correlation accuracy measures how well system, application, and network logs align to provide a comprehensive attack timeline, while forensic analysis effectiveness evaluates the system's ability to support security teams in post-incident analysis by delivering actionable insights. Together, these evaluations ensure the system's robustness, accuracy, and practical applicability in real-world environments.

This methodology integrates data collection (normal and malicious-known traffic), feature extraction, supervised classification, and anomaly detection into a cohesive framework. By training a binary classifier on labeled attack data and pairing it with an anomaly detector trained solely on normal traffic, the hybrid pipeline aims to achieve robust performance against both established and emergent attack vectors. Through iterative feedback and retraining, the system refines its detection capabilities over time, addressing the evolving nature of web-based threats.

3.7 RESEARCH TIMELINE

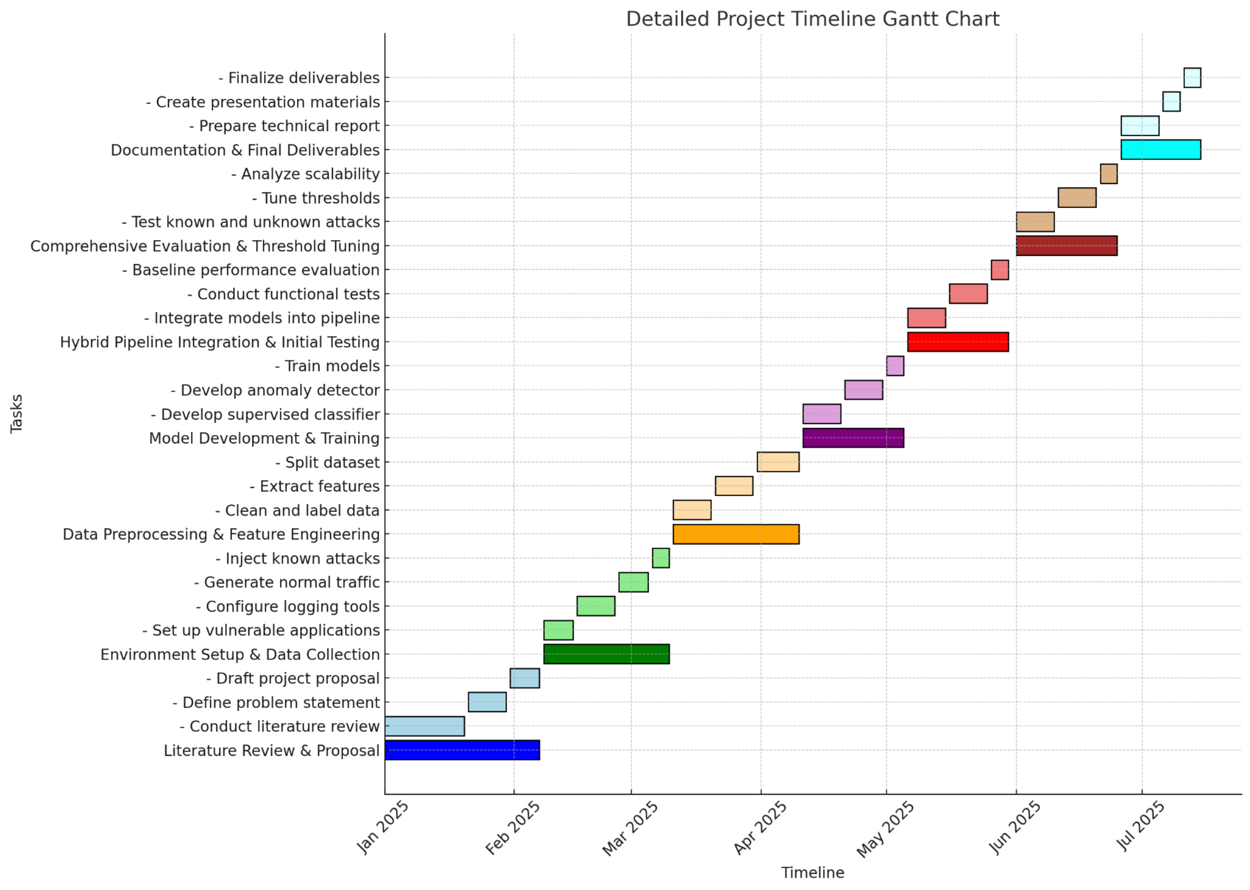

Figure 3: Research Timeline

3.8. CONCLUSION

This research proposal outlines a hierarchical, multistage intrusion detection framework focused on application-layer threats, a critical yet often under-addressed aspect of modern cybersecurity. By combining supervised classification for known malicious patterns with anomaly detection for novel or zero-day exploits, the proposed system aims to improve accuracy, scalability, and adaptability beyond what traditional rule-based Intrusion Detection Systems and Web Application Firewalls can offer.

- Accuracy is enhanced through the targeted use of machine learning and deep learning models, capitalizing on labeled attack data for supervised tasks and one-class methods for anomaly detection.
- Scalability is managed by carefully selecting efficient ML algorithms, leveraging containerized or virtualized test environments, and exploring cloud-based resources for large-scale traffic simulation and model training.
- Adaptability stems from a feedback loop, wherein newly discovered attack patterns or benign anomalies are fed back into the training set, ensuring the system evolves alongside changing threat landscapes.
- Component Identification improves forensic capabilities by pinpointing affected microservices or functions, reducing response time and enabling rapid mitigation efforts.

By integrating seamlessly into existing WAFs and API gateways, this solution seeks to minimize overhead while maximizing security benefits for web applications and APIs. Future work may extend toward adversarial defense strategies, continuous learning, and big data frameworks to further enhance detection capabilities in real-world high-throughput scenarios. Ultimately, the project aims to deliver a robust, efficient, and forward-thinking approach to application-layer intrusion detection, with a unique emphasis on identifying and mitigating vulnerabilities at the component level, significantly contributing to the broader domain of web security.

3.9. REFERENCES

[1] Adek, R. T., and Ula, M. (2020), “A Survey on The Accuracy of Machine Learning Techniques for Intrusion and Anomaly Detection on Public Data Sets,” in 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), Medan, Indonesia: IEEE, Jul. 2020, pp. 19–27, doi: 10.1109/DATABIA50434.2020.9190436.

[2] Applebaum, S., Gaber, T., and Ahmed, A. (2021), “Signature-based and Machine-Learning-based Web Application Firewalls: A Short Survey,” Procedia Comput. Sci.,vol. 189, pp. 359–367, doi: 10.1016/j.procs.2021.05.105.

[3] Bao, K., and Ding, Y. (2020), “Network security analysis using big data technology and improved neural network,” J. Ambient Intell. Humaniz. Comput., May 2020, doi: 10.1007/s12652-020-02080-1.

[4] Betarte, G., Gimenez, E., Martinez, R., and Pardo, A. (2018a), “Improving Web Application Firewalls through Anomaly Detection,” in 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec. 2018, doi: 10.1109/ICMLA.2018.00124.

[5] Betarte, G., Gimenez, E., Martinez, R., and Pardo, A. (2018b), “Improving Web Application Firewalls through Anomaly Detection,” in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA: IEEE, Dec. 2018, pp. 779–784, doi: 10.1109/ICMLA.2018.00124.

[6] Betarte, G., Giménez, E., Martínez, R., and Pardo, Á. (2018c), “Machine learning-assisted virtual patching of web applications,” arXiv, Mar. 2018. [Online]. Available: https://arxiv.org/abs/1803.05529.

[7] Betarte, G., Pardo, A., and Martinez, R. (2018d), “Web application attacks detection using machine learning techniques,” in 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), Dec. 2018, pp. 1065–1072, doi: 10.1109/ICMLA.2018.00174.

[8] Betarte, G., Pardo, A., and Martinez, R. (2018e), “Web Application Attacks Detection Using Machine Learning Techniques,” in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA: IEEE, Dec. 2018, pp. 1065–1072, doi: 10.1109/ICMLA.2018.00174.

[9] Buczak, A. L., and Guven, E. (2016), “A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection,” IEEE Commun. Surv. Tutorials, vol. 18, no. 2, pp. 1153–1176, doi: 10.1109/COMST.2015.2494502.

[10] Chen, M. (2023), Compound Prediction Model of Information Network Security Situation Based on Support Vector Machine and Particle Swarm Optimization Algorithm, pp. 1–6, doi: 10.1109/icaisc58445.2023.10200752.

[11] Chen, W.-P., Ao, Z.-G., Tu, Y.-Q., Kang, X.-D., and Zhao, Z.-N. (2017), “Network Situation Awareness Model Prediction Method Based on Genetic Optimization Support Vector Machine,” Advances in Computer Science Research (ACRS), vol. 54, Jan. 2017, doi: 10.2991/cnct-16.2017.69.

[12] Cheng, J., Lin, B., Wei, J., and Xia, A. (2022), "The Compound Prediction Analysis of Information Network Security Situation based on Support Vector Combined with BP Neural Network Learning Algorithm," vol. 16, 2022.

[13] Dawadi, B., Adhikari, B., and Srivastava, D. (2023), "Deep Learning Technique-Enabled Web Application Firewall for the Detection of Web Attacks," Sensors, vol. 23, no. 4, p. 2073, Feb. 2023, doi: 10.3390/s23042073.

[14] Ferrag, M. A., Maglaras, L., Moschoyiannis, S., and Janicke, H. (2020), "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," J. Inf. Secur. Appl., vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.

[15] Fu, H., Liu, M., and Zou, Y. (2023), "Simulation of Network Security Situation Assessment Model Based on Machine Learning Algorithm," in 2023 International Conference on Power, Electrical Engineering, Electronics and Control (PEEEC), Athens, Greece: IEEE, Sep. 2023, pp. 402–407, doi: 10.1109/PEEEC60561.2023.00084.

[16] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., and Ahmad, R. (2022), "Machine Learning and Deep Learning Approaches for CyberSecurity: A Review," IEEE Access, vol. 10, pp. 19572–19585, doi: 10.1109/ACCESS.2022.3151248.

[17] Hao, G., and Xianyu, J. (2022), "Short-term load forecasting based on improved manta ray algorithm to optimize neural network," J. Phys.: Conf. Ser., vol. 2189, no. 1, p. 012019, Feb. 2022, doi: 10.1088/1742-6596/2189/1/012019.

[18] Kim, A., Park, M., and Lee, D. H. (2020), "AI-IDS: Application of Deep Learning to Real-Time Web Intrusion Detection," IEEE Access, vol. 8, pp. 70245–70261, doi: 10.1109/ACCESS.2020.2986882.

[19] Khan, A. W., Zaib, S., Khan, F., Tarimer, I., Seo, J. T., and Shin, J. (2022), "Analyzing and evaluating critical cyber security challenges faced by vendor organizations in software development: SLR based approach," IEEE Access, vol. 10, pp. 65044–65054, Jan. 2022, doi: 10.1109/ACCESS.2022.3179822.

[20] Li, Z., Ma, T., Zhou, Y., and Wang, X. (2021), "Research and simulation of network security situation prediction algorithm," J. Phys.: Conf. Ser., vol. 1941, no. 1, p. 012051, Jun. 2021, doi: 10.1088/1742-6596/1941/1/012051.

[21] Mokbal, F. M. M., Dan, W., Imran, A., Jiuchuan, L., Akhtar, F., and Xiaoxi, W. (2019), "MLPXSS: An Integrated XSS-Based Attack Detection Scheme in Web Applications Using Multilayer Perceptron Technique," IEEE Access, vol. 7, pp. 100567–100580, doi: 10.1109/ACCESS.2019.2927417.

[22] Montes, N., Betarte, G., Martínez, R., and Pardo, A. (2021), "Web application attacks detection using deep learning," in Lecture Notes in Computer Science, pp. 227–236, doi: 10.1007/978-3-030-93420-0_22.

[23] Praseed, A., and Thilagam, P. S. (2019), “DDoS Attacks at the Application Layer: Challenges and Research Perspectives for Safeguarding Web Applications,” IEEE Commun. Surv. Tutorials, vol. 21, no. 1, pp. 661–685, doi: 10.1109/COMST.2018.2870658.

[24] Saha, A., and Sanyal, S. (2024), “Application layer intrusion detection with combination of Explicit-Rule-based and Machine learning algorithms and deployment in cyber-defence program,” Int. J. Adv. Netw. Appl., Nov. 2024. [Online]. Available: https://www.researchgate.net/publication/268226901.

[25] Shar, L. K., Briand, L. C., and Tan, H. B. K. (2015), “Web Application Vulnerability Prediction Using Hybrid Program Analysis and Machine Learning,” IEEE Trans. Dependable Secure Comput., vol. 12, no. 6, pp. 688–707, Nov. 2015, doi: 10.1109/TDSC.2014.2373377.

[26] Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018), “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization,” 2018.

[27] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S. (2019), “Deep Learning Approach for Intelligent Intrusion Detection System,” IEEE Access, vol. 7, pp. 41525–41550, doi: 10.1109/ACCESS.2019.2895334.

[28] Ye, W., Wang, H., and Zhong, Y. (2022), “Optimization of network security protection situation based on data clustering,” Int. J. Syst. Assur. Eng. Manag., Feb. 2022, doi: 10.1007/s13198-021-01529-6.

[29] Yu, Y. (2024), “A network security situation assessment method based on fusion model,” Discov. Appl. Sci., vol. 6, no. 3, p. 97, Feb. 2024, doi: 10.1007/s42452-024-05723-6.

[30] Zhang, R., Liu, M., Pan, Z., and Yin, Y. (2022), “Network Security Situation Assessment based on improved WOA-SVM,” IEEE Access, vol. 10, pp. 96273–96283, Jan. 2022, doi: 10.1109/ACCESS.2022.3204663.