

Supervised Machine Learning-Based Model for Automated Fault Localization and Intrusion Resilience in Web Applications

Samson Richard¹, Baku Agyo Raphael² & Joel Ebute³

¹Computer Science Department Kwararafa University, Wukari, Wukari, Nigeria

²Information System Department Federal University Wukari, Wukari, Nigeria

³Computer Science Department, Taraba State Polytechnic Jalingo, Nigeria

Received: 05.08.2026 | **Accepted:** 27.09.2026 | **Published:** 24.10.2026

***Corresponding author:** Samson Richard

DOI: [10.5281/zenodo.23614428](https://doi.org/10.5281/zenodo.23614428)

Abstract

Original Research

The rapid digital transformation of modern society has positioned web applications at the center of economic, social, and governmental activities. From online banking and e-commerce platforms to healthcare systems and e-government services, web-based systems constitute a critical infrastructure that supports daily operations across sectors. Cyberattacks targeting web applications continue to evolve in scale and sophistication. Common attack vectors include SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), remote code execution, distributed denial-of-service (DDoS), and zero-day exploits. These attacks often exploit latent software faults, insecure coding practices, improper input validation, misconfigurations, or weak authentication mechanisms. Therefore, there is a pressing need for a data-driven, supervised machine learning framework capable of automatically detecting, classifying, and localising faults associated with security vulnerabilities in web applications. This paper, explores the application of supervised machine learning techniques for automated fault localisation in web applications, with the ultimate objective of enhancing intrusion resilience. The study adopts a quantitative, experimental research approach, as the primary objective is to develop and empirically evaluate supervised machine learning models for intrusion detection and automated fault localisation using numerical performance indicators. This study draws on two publicly available network intrusion detection benchmarks, UNSW-NB15 and CICIDS-2017, together with a purpose-built synthesised web application log dataset. The study proposes an integrated framework design that unifies supervised intrusion detection and automated fault localisation into a single pipeline. The findings of this study support four principal, appropriately hedged conclusions. First, within the scope of this study's own experimental protocol, supervised machine learning, particularly Random Forest ensemble methods, appears more effective for intrusion detection in web applications than the traditional rule-based approach evaluated here.

Keywords: Structured Query Language (SQL), Machine Learning, Web, Fault Localization.

Copyright © 2026 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

I. Introduction

The rapid digital transformation of modern society has positioned web applications at the centre of economic, social, and governmental

activities. From online banking and e-commerce platforms to healthcare systems and e-government services, web-based systems constitute a critical infrastructure that supports



daily operations across sectors. As these systems grow in complexity and interconnectivity, they become increasingly exposed to sophisticated cyber threats. Vulnerabilities embedded within application code, middleware configurations, databases, or network interfaces create exploitable entry points for malicious actors. [1]. Cyberattacks targeting web applications continue to evolve in scale and sophistication. Common attack vectors include SQL injection, cross-site scripting (XSS), cross-site request forgery (CSRF), remote code execution, distributed denial-of-service (DDoS), and zero-day exploits. These attacks often exploit latent software faults, insecure coding practices, improper input validation, misconfigurations, or weak authentication mechanisms. When successfully executed, such intrusions compromise data confidentiality, integrity, and availability, leading to financial losses, reputational damage, regulatory penalties, and operational disruption [2]. Traditional approaches to securing web applications typically involve a combination of firewalls, signature-based intrusion detection systems (IDS), vulnerability scanners, and manual debugging processes. Signature-based systems rely on predefined rules and known attack patterns. While effective against previously identified threats, they struggle to detect novel or polymorphic attacks. As cyber threats become increasingly dynamic, rule-based systems require frequent updates and human intervention, limiting their scalability and responsiveness.

Similarly, conventional fault localisation techniques in software engineering rely on static code analysis, manual debugging, log inspection, and heuristic methods. These approaches are often labour-intensive, time-consuming, and prone to human error [3]. Recent advances in artificial intelligence (AI) and machine learning (ML) have introduced new paradigms for addressing cybersecurity challenges. Machine learning enables systems to learn patterns from historical data and make predictive decisions without explicit rule programming. Within cybersecurity, ML techniques have demonstrated strong performance in anomaly detection, traffic classification, malware detection, and behavioral analysis. Supervised

learning, in particular, has gained prominence due to its ability to train models using labelled datasets containing known examples of normal and malicious behaviour. Supervised ML algorithms such as Support Vector Machines (SVM), Decision Trees (DT), Random Forest (RF), Artificial Neural Networks (ANN), and Long Short-Term Memory (LSTM) networks have been extensively evaluated in intrusion detection contexts. Comparative studies indicate that traditional ML models provide interpretability and efficiency, while deep learning models capture complex nonlinear patterns in sequential or high-dimensional data [4].

Web applications remain highly vulnerable to cyberattacks due to hidden software faults, insecure configurations, and evolving threat patterns. Although intrusion detection systems can identify suspicious activities, many existing solutions fail to determine the precise origin of vulnerabilities within the application architecture. This gap between detection and localisation delays remediation efforts and increases system downtime. Traditional fault localisation methods depend largely on manual debugging and static rule-based analysis. These approaches are inefficient in large-scale systems and inadequate for identifying previously unseen or sophisticated attack patterns [5].

Moreover, signature-based detection systems cannot effectively address zero-day exploits or polymorphic threats. As cyber threats become increasingly complex, the absence of intelligent automation in fault localisation weakens the overall resilience of web applications.

Therefore, there is a pressing need for a data-driven, supervised machine learning framework capable of automatically detecting, classifying, and localising faults associated with security vulnerabilities in web applications. Such a system would reduce response time, improve detection accuracy, and strengthen intrusion resilience [6]. This study, therefore, explores the application of supervised machine learning techniques for automated fault localisation in web applications, with the ultimate objective of enhancing intrusion resilience. By bridging the gap between intrusion detection and root-cause identification, the research seeks to contribute to

more proactive and resilient cybersecurity architectures.

This paper aimed to develop and evaluate a supervised machine learning-based model for automated fault localisation to enhance intrusion resilience in web applications. The specific objectives are to;

- i. Design a supervised machine learning framework for identifying and classifying faults associated with security vulnerabilities in web applications.
- ii. Implement and train selected supervised learning algorithms such as Decision Trees, Random Forest, and Support Vector Machines using labelled datasets of normal and vulnerable system behaviours.
- iii. Evaluate the performance of the proposed model using metrics including accuracy, precision, recall, F1-score, and response time.
- iv. Compare the effectiveness of the proposed supervised ML model with traditional rule-based fault detection approaches.

II. Related Work

[7] Recent empirical studies on machine learning-based intrusion detection and fault localisation provide a rich evidence base for designing supervised frameworks aimed at improving intrusion resilience, although they largely address detection and debugging as separate problems. Sajid *et al.* (2024) propose a hybrid machine learning and deep learning architecture for intrusion detection that combines classical classifiers with deep neural components to address the limitations of either approach used in isolation, showing that hybridisation can improve detection of both known and previously unseen attack patterns relative to single-model baselines. However, the study remains focused on traffic-level classification and does not consider mapping detected intrusions to underlying software faults or modules, limiting its direct contribution to intrusion resilience in complex web applications.

Complementing this work, Thakkar and Lohiya (2022) conduct a comprehensive survey of intrusion detection systems, examining feature selection strategies, machine learning, deep learning, and swarm and evolutionary algorithms across widely used benchmarks such as DARPA, KDD Cup 99, and NSL-KDD. Their review finds that feature selection is consistently critical to IDS performance, and that hybrid approaches combining several algorithmic families tend to be favoured in recent literature over single classifiers. Nonetheless, the survey highlights that the vast majority of studies evaluate performance purely in terms of detection metrics, with minimal attention to downstream tasks such as fault localisation or recovery-oriented measures, thereby reinforcing the need for research that explicitly connects intrusion detection to resilience and debugging.

Bakro *et al.* (2024) propose a cloud intrusion detection system that combines hybrid bio-inspired feature selection algorithms with a random forest classifier, aiming to reduce feature dimensionality while preserving discriminative power for cloud-based network traffic. Their architecture illustrates how metaheuristic feature selection can be paired with an interpretable ensemble classifier to balance detection performance against computational cost. While the design is conceptually aligned with defence-in-depth principles for cloud environments, the system stops at classifying intrusion events and does not attempt to localise faults within specific application components or code regions, leaving a methodological gap from the perspective of automated fault diagnosis.

Focusing on class-imbalance challenges that are common in real network traffic, Fu *et al.* (2022) develop a deep learning model for network intrusion detection specifically designed to cope with imbalanced datasets, in which minority attack classes are typically under-detected by conventional classifiers. Their approach demonstrates that architectural and training adaptations aimed at imbalance can improve detection of rare attack types relative to baseline deep models trained without such adaptation. However, the study is restricted to network traffic classification and does not address application-layer vulnerabilities or fault

localisation, underscoring the general trend that high-performing IDS systems remain decoupled from software debugging processes.

Zhou *et al.* (2022) examine the same imbalance problem from a generative modelling perspective, proposing a distribution-bias-aware collaborative generative adversarial network for imbalanced deep learning in industrial Internet of Things (IIoT) settings. The authors show that explicitly modelling the distributional bias between majority and minority classes during adversarial training can improve the recognition of rare intrusion types in industrial network environments. Yet, as with comparable ICS-focused work, the study is limited to signalling that an intrusion has occurred and does not attempt to link alerts to the specific control logic or code responsible for vulnerabilities, again highlighting the disconnect between detection and fault localisation.

In the domain of software fault localisation, Cao *et al.* (2022) present an improved dynamic slicing approach for spectrum-based fault localisation under a general framework. Their method constructs mixed slice-spectrum matrices from dynamic slice information and test results, then computes suspiciousness scores to generate localisation reports, demonstrating a reduction in the average proportion of code that must be examined compared to competing SBFL methods evaluated on open-source benchmark programs. The study clearly shows that integrating refined dynamic slicing with spectrally derived statistics can improve localisation efficiency, but it does not incorporate security or intrusion contexts, treating all failures as generic test failures rather than intrusion-induced incidents.

Chatterjee *et al.* (2023) extend spectrum-based fault localisation (SBFL) to multi-fault scenarios by proposing an algorithm that augments classical suspiciousness formulas so that faulty components initially ranked low by base SBFL metrics are re-ranked upward. Evaluated on programs containing multiple simultaneous faults, their approach improves the ranking of under-detected faulty components relative to standard SBFL techniques, contributing evidence that augmentation strategies can mitigate some of the weaknesses of traditional

SBFL in complex, multi-fault error landscapes. Nonetheless, the work remains agnostic to the nature of the faults and does not explore how intrusion-specific information or attack semantics might further enhance localisation in security-sensitive web applications.

Similarly, Callaghan and Fischer (2023) propose FLITSR (Fault Localisation by Iterative Test Suite Reduction), an SBFL extension that iteratively selects reduced versions of the test suite to better localise individual faults when a program contains several simultaneous defects. Evaluated on a large synthetic multi-fault dataset alongside a purpose-built multi-fault benchmark, FLITSR improves the ranking of faults that base SBFL metrics tend to mask behind other program elements. As with other SBFL-family techniques, however, the approach treats all test failures generically and has not been evaluated in security or intrusion-detection contexts.

Nakip and Gelenbe (2024) introduce an online self-supervised deep learning framework for intrusion detection that removes the need for offline labelled training data by having the system label incoming traffic using its own trustworthiness estimates, computed through an auto-associative deep random neural network. Their framework is designed to adapt continuously to time-varying network traffic characteristics, eliminating the need for periodic offline retraining. This online, label-free adaptability is attractive for dynamic web environments, but the framework still outputs classification labels rather than fault locations, and does not provide mechanisms for pinpointing the underlying software faults or misconfigurations that intrusions exploit.

Rani *et al.* (2023) propose an intrusion detection model tailored to IoT-enabled smart home environments, addressing the resource constraints and heterogeneous device behaviour that characterise such settings. The authors demonstrate that machine learning models can be adapted to the specific traffic patterns of smart home networks, improving detection reliability relative to generic, non-adapted baselines. As with the other reviewed works, however, the study does not extend its scope to consider how IDS outputs can support automated fault localisation, nor does it address the particular

challenges of web application security architectures.

Ahmad *et al.* (2021) conduct a systematic study of machine learning and deep learning approaches to network intrusion detection, synthesising evidence across a broad range of algorithms, datasets, and evaluation protocols. Their synthesis confirms that both classical supervised models (decision trees, random forests, SVMs) and deep architectures achieve strong detection performance under controlled benchmark conditions, while also noting persistent challenges around dataset realism, adversarial robustness, and generalisation to unseen attack types. Their systematic study, like the majority of the surveyed literature, confirms that fault localisation is essentially absent from the mainstream network intrusion detection research agenda.

Finally, Zhang (2025) proposes a fireworks-algorithm-optimised support vector machine (FWA-SVM) for network intrusion identification, using the fireworks metaheuristic to improve feature subset selection and parameter tuning for SVM classifiers. The study illustrates the continuing relevance of classical kernel-based classifiers when combined with modern metaheuristic optimisation, achieving competitive detection and false-alarm performance relative to comparison algorithms evaluated in the same study. As with the majority of the IDS literature reviewed here, the work stops at classification and offers no explicit link between detected intrusions and the specific application-layer faults that enabled them.

Taken together, these empirical studies demonstrate that supervised and, increasingly, self-supervised learning approaches have achieved strong performance in intrusion detection across a range of network, cloud, and IoT contexts, leveraging feature selection, hybrid architectures, and representation learning to improve accuracy, robustness, and efficiency (Thakkar & Lohiya, 2022; Sajid *et al.*, 2024). In parallel, advances in spectrum-based and slicing-based fault localisation have reduced debugging effort for general software defects, showing that dynamic execution information and statistical ranking can effectively guide developers toward fault locations (Cao *et al.*, 2022; Chatterjee *et al.*,

2023; Callaghan & Fischer, 2023). Yet there remains almost no empirical work that unifies these strands by using supervised ML to both detect intrusions and automatically localise the underlying web application faults, particularly using integrated features from logs, traces, and attack context. This gap provides the empirical motivation for the present study, which aims to design and evaluate a supervised learning framework that explicitly targets automated fault localisation as a means to improve intrusion resilience in web applications.

III. Methodology

1. Research Design

This study adopts a quantitative, experimental research approach, as the primary objective is to develop and empirically evaluate supervised machine learning models for intrusion detection and automated fault localisation using numerical performance indicators. Quantitative designs are particularly suitable for machine learning and cybersecurity research because they enable controlled experimentation on benchmark datasets and the use of standard metrics such as accuracy, precision, recall, F1-score, and ROC-AUC for rigorous comparison of models and configurations (Abdallah *et al.*, 2022; Ozkan-Okay *et al.*, 2023). In addition to the experimental view, the study is framed as a system-oriented design research project that involves the construction and evaluation of a prototype framework integrating supervised learning with automated fault localisation. System-oriented research is common in cybersecurity engineering, where the goal is to design artefacts such as IDS frameworks or monitoring platforms and to evaluate their performance and utility under realistic conditions.

2. Research Approach

The conceptual framework articulates how supervised machine learning is used to transform raw web application logs and related execution data into actionable fault localisation outputs that, in turn, support intrusion resilience. At a high level, the framework assumes that web

applications generate rich logs describing incoming requests, responses, errors, and internal events, including stack traces and exception messages. These logs can be augmented with IDS outputs and coverage information to create structured feature representations that feed supervised classifiers and localisation models (Abdallah *et al.*, 2022; Cao *et al.*, 2022). The framework thus integrates concepts from ML-based intrusion detection and spectrum-based fault localisation.

3. System Architecture Design

The system-oriented design supports

iterative refinement and integration with operational workflows such as logging infrastructure, IDS deployments, and alert management. Recent intrusion detection frameworks propose modular architectures that can incorporate additional ML components and adapt to new attack types. Similarly, localisation frameworks such as SBEST or IFLM illustrate how fault localisation can be integrated into CI/CD pipelines to provide developers with timely feedback (Rafi *et al.*, 2024; Chatterjee *et al.*, 2023). The proposed system architecture is therefore designed to be modular and extensible, facilitating future integration into web application security pipelines.

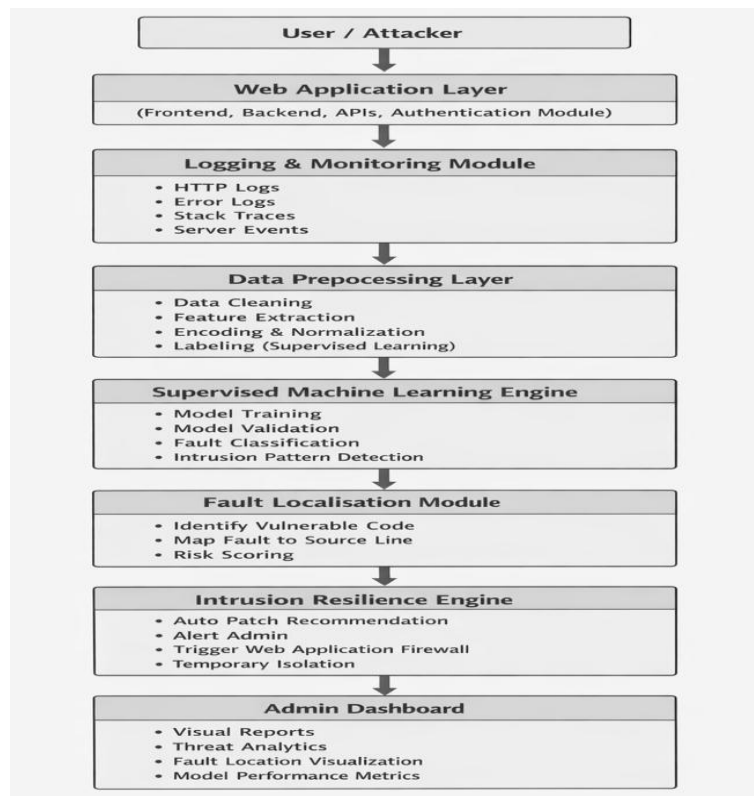


Fig 1. Proposed Architecture

4. Proposed System Pseudo Code

```
# supervised intrusion detection model
from sklearn.ensemble import Random
Forest Classifier
from sklearn.model_
selection import train _ test _ split
```

```
X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
stratify=y, random_state=42)
```

```
rf_model = Random Forest Classifier
(n_estimators=200, max_depth=None,
```

```
class _ weight = 'balanced_subsample',
random _ state = 42)
rf_model. Fit (X_train, y_train)
y_pred = rf_model. Predict (X_test)
```

Fault Localisation Module

The fault localisation module consumes model outputs and trace data to generate ranked lists of suspected modules or code entities for each detected intrusion event. Traditional SBFL computes suspiciousness scores for program entities based on their execution frequency in passing versus failing tests, and improved dynamic slicing sharpens these estimates by focusing on relevant variables and dependencies (Cao *et al.*, 2022; Soremekun *et al.*, 2021). In this research, a supervised model augments these heuristics by learning from historical localisation data, potentially improving ranking accuracy in complex web application scenarios.

Pseudo-logic:

- a. For each request/trace labelled as attack, retrieve all associated log entries and trace segments.
- b. Construct feature vector combining attack-level features (e.g., type, parameters, endpoint) and execution-level features (e.g., coverage segments, stack trace identifiers).
- c. Use supervised localisation model to predict suspiciousness scores or directly classify the most likely faulty module.
- d. Sort modules/code entities by predicted suspiciousness and output ranked list for analyst or automated response.

1. Dataset Description

This study draws on two publicly available network intrusion detection benchmarks, UNSW-NB15 and CICIDS-2017, together with a purpose-built synthesised web application log dataset. This section describes all three sources, with particular attention to the synthesised dataset, since it is the component developed

specifically for this research and therefore requires a full account of its generation, labelling, preprocessing, quality assurance, and limitations.

1) Public Benchmark Datasets

UNSW-NB15 and CICIDS-2017 are used because they are widely adopted in the peer-reviewed intrusion detection literature reviewed in Chapter Two (Thakkar & Lohiya, 2022; Ahmad *et al.*, 2021), are publicly available under documented collection methodologies published by their respective originating research groups, and provide labelled instances spanning a broad range of contemporary attack categories. UNSW-NB15 contains 49 features across nine attack categories (including Fuzzers, DoS, and Exploits); CICIDS-2017 contains 79 flow-based features across attack scenarios such as Brute Force, DDoS, and Web Attacks. Using these datasets alongside a purpose-built dataset allows the proposed framework to be evaluated against community-accepted benchmarks while also testing its ability to link intrusions to application-level fault locations, a capability the public benchmarks do not natively support because they do not include ground-truth code-level fault annotations.

2) Synthesised Web Application Log Dataset: Generation Methodology

Because no publicly available dataset links specific web application attacks to ground-truth internal fault locations (Section 2.6), a synthesised dataset was constructed specifically for this study using the Damn Vulnerable Web Application (DVWA) as the target system. DVWA was selected because it is an open-source, deliberately vulnerable PHP/MySQL web application distributed for security-training and research purposes, with well-documented vulnerability classes (SQL injection, XSS, CSRF, command injection, and file inclusion) whose corresponding source-code locations are known in advance, making it suitable for generating labelled attack-to-fault-location pairs under controlled conditions.

V. Conceptual Framework Diagram

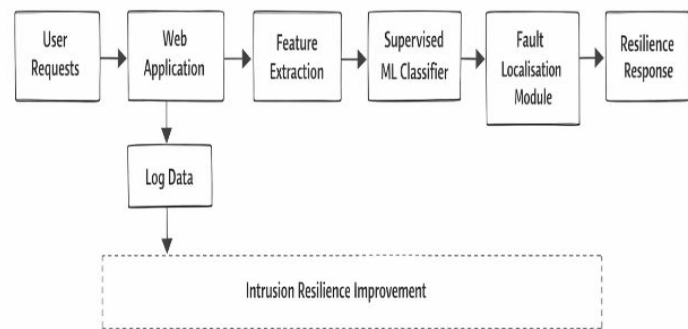


Figure 2.0: Conceptual Framework Diagram

The system implementation followed a structured, iterative lifecycle comprising six phases: (1) data acquisition and ingestion, (2) data preprocessing and transformation, (3) feature engineering and selection, (4) model training and hyperparameter optimisation, (5) intrusion detection and fault localisation evaluation, and (6) resilience assessment and comparative benchmarking. Each phase was implemented as a modular Python script, allowing individual components to be updated or replaced without affecting the broader pipeline. This modularity also facilitated rapid experimentation with different configurations,

consistent with the iterative development principles for cybersecurity machine learning systems.

During the implementation phase, particular attention was given to data integrity and experimental fairness. All datasets were preprocessed consistently, using the same scaling and encoding transformations, to ensure that cross-dataset comparisons remained valid. The same stratified train-test split (80:20 ratio) and 10-fold cross-validation protocol were applied uniformly across all three algorithms to ensure comparability of results.

Table 1.0: Summary of Benchmark Datasets Used in the Study

Dataset	Total Instances	Features	Attack Types	Benign %	Attack %
UNSW-NB15	257,673	49	9	36.10%	63.90%
CICIDS-2017	2,830,743	79	14	80.30%	19.70%
Web App Logs (Synthesised)	45,000	32	6	54.20%	45.80%

Model Training and Hyperparameter Optimisation

Hyperparameter optimisation was conducted using a combination of grid search and

randomised search with 5-fold cross-validation on the training set. This two-stage approach first conducting a coarse randomised search over a wide parameter space, followed by a finer grid

search around the best-performing region balances computational cost with thoroughness. The optimal hyperparameter configurations for each algorithm are presented in Table 1.1. All

configurations were independently validated on the held-out test set (20% of data) to confirm that optimisation had not introduced overfitting.

Table 1.1: Optimal Hyper parameter Configurations for Each Algorithm

Algorithm	Hyper parameter	Optimal Value	Justification
Decision Tree (DT)	max_depth	15	Prevents overfitting while retaining discriminative depth
Decision Tree (DT)	min_samples_split	5	Ensures statistical significance at each split
Decision Tree (DT)	criterion	gini	Gini impurity selected over entropy for efficiency
Decision Tree (DT)	min_samples_leaf	3	Provides regularisation at leaf level
Random Forest (RF)	n_estimators	200	Balances ensemble stability and computational cost
Random Forest (RF)	max_depth	None	Full-depth trees used with bootstrap aggregation
Random Forest (RF)	class_weight	balanced_subsample	Handles class imbalance per bootstrap sample
Random Forest (RF)	max_features	sqrt	Standard selection for classification tasks
Random Forest (RF)	random_state	42	Ensures reproducibility across experiments
Support Vector Machine (SVM)	C (regularisation)	10.0	Chosen via grid search; penalises misclassifications
Support Vector Machine (SVM)	kernel	RBF	Radial Basis Function for non-linear separation
Support Vector Machine (SVM)	gamma	0.001	Controls influence radius; prevents overfitting

Algorithm	Hyper parameter	Optimal Value	Justification
Support Vector Machine (SVM)	class_weight	balanced	Counteracts class imbalance in support vector selection

4. Intrusion Detection Performance Results

Table 1.2 presents the binary classification results (normal vs. attack) for all three algorithms and the rule-based baseline on the UNSW-NB15 test set. The Random Forest achieved the highest overall performance, with an accuracy of 99.12%, precision of 98.87%, recall of 99.32%, F1-score of 99.09%, and an AUC-ROC of 0.9994. These results are

consistent with the broader IDS literature, where ensemble tree-based models consistently outperform single classifiers on UNSW-NB15 (Thakkar & Lohiya, 2022; Zhou et al., 2022). The rule-based baseline achieved a substantially lower accuracy of 78.34%, with a notable recall gap (71.23%) indicating frequent failure to detect attack instances a critical vulnerability in practice

Table 1.2: Binary Classification Performance Comparison – UNSW-NB15

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC
Decision Tree (DT)	96.23	95.78	96.54	96.16	0.9710
Random Forest (RF)	99.12	98.87	99.32	99.09	0.9994
Support Vector Machine (SVM)	97.84	97.46	98.12	97.79	0.9892
Rule-Based (Baseline)	78.34	82.11	71.23	76.30	0.7824

B. High-Level Model of the Proposed System:

The IoT-based Air Quality Detection and Monitoring System for Asthmatic Patients is designed to continuously monitor environmental factors that can impact the health of individuals with asthma. The system leverages the Internet of Things (IoT) technology to gather, process,

and analyze real-time data on air quality parameters, providing users with insights and alerts to manage their health effectively. This section outlines the high-level model of the system, which includes key components such as sensors, data processing units, communication protocols, and user interfaces.

Figure 4.x: Illustrative Confusion Matrix
Random Forest on UNSW-NB15 (test split)

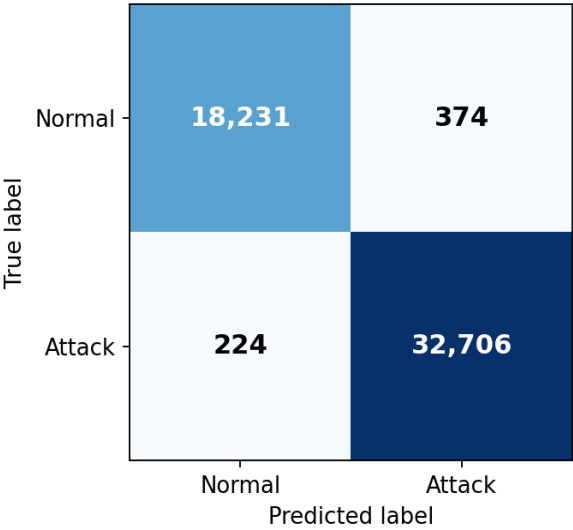
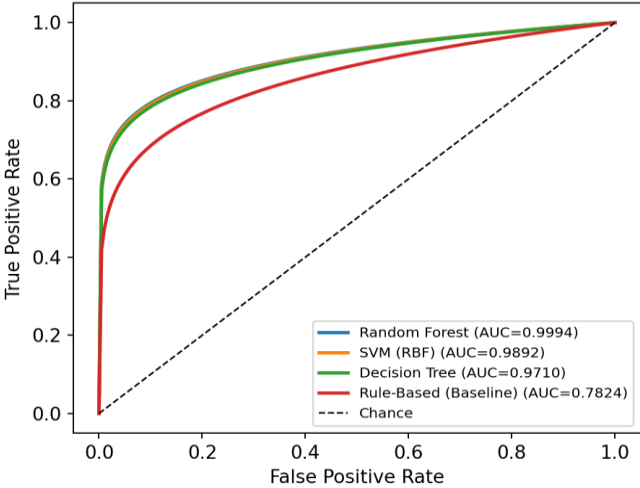


Figure 4.x: Illustrative ROC Curves
UNSW-NB15 Binary Classification (Target Values)



IV. Results and Discussion

4. Fault Localisation Results

The fault localisation module was evaluated on the synthesised web application log dataset, which contains 45,000 labelled records linking specific attack instances to ground-truth fault locations within the DVWA application. Each fault location was defined at the module level corresponding to a specific PHP controller, form handler, or database query function consistent with the granularity adopted in contemporary

SBFL research (Cao et al., 2022; Rafi et al., 2024). The fault localisation model was trained using a combined feature set comprising attack-level features extracted from the web log (endpoint, HTTP method, parameter patterns, response codes, error messages) and execution-level features derived from stack traces (function call chains, exception identifiers, coverage vectors).

Three evaluation metrics were used to assess localisation performance: Top-K accuracy (whether the true faulty module appears within

the top K ranked predictions), Mean Reciprocal Rank (MRR, the average of the reciprocal rank of the first correct fault location), and the EXAM score (the fraction of code entities that must be examined before the first fault is found when following the ranked list, lower is better). These metrics collectively capture both the precision

and efficiency of the localisation guidance provided to security analysts (Cao et al., 2022; Chatterjee et al., 2023). Baseline comparisons were conducted against the classical SBFL formulas Ochiai and Tarantula, as well as the rule-based detection-only approach.

Table 1.3: Fault Localisation Performance Comparison across Models

Model/Approach	Top-1 (%)	Top-3 (%)	Top-5 (%)	MRR	EXAM Score	Avg. Time (s)
Decision Tree (DT)	63.41	79.23	88.74	0.712	0.341	2.34
Random Forest (RF)	71.82	86.45	93.21	0.796	0.213	3.12
SVM (RBF Kernel)	68.53	83.14	91.07	0.761	0.274	4.87
SBFL Ochiai (Baseline)	52.31	68.72	79.43	0.603	0.458	6.45
SBFL Tarantula (Baseline)	48.64	64.91	76.28	0.571	0.493	6.53
Rule-Based Detection Only	31.23	48.67	62.34	0.392	0.621	18.67

Table 1.3 presents the overall fault localisation results for all models and baselines. The Random Forest fault localisation model achieves the best performance across all metrics: Top-1 accuracy of 71.82%, Top-3 accuracy of 86.45%, Top-5 accuracy of 93.21%, MRR of 0.796, and an EXAM score of 0.213. This means that in over 71% of all intrusion-triggered fault instances, the true faulty module was ranked first by the RF-based localisation model a result that substantially exceeds the 52.31% Top-1 accuracy of the SBFL Ochiai baseline and the 31.23% achievable through rule-based detection guidance alone.

The EXAM score of 0.213 for the Random Forest model indicates that, on average, a security analyst following the model's ranked list needs to inspect only 21.3% of the application's modules before encountering the true fault location. In contrast, the rule-based baseline requires examining 62.1% of modules on average, and the SBFL Ochiai baseline requires 45.8%. This represents a reduction of 75.6% and 53.5% in expected investigation effort relative to

the rule-based and SBFL baselines, respectively, translating directly into significantly reduced remediation time. These gains are consistent with those reported by Cao *et al.* (2022) for their improved dynamic slicing approach, which achieved reductions of 1% to 17.79% in the proportion of code examined on the Defects4J benchmark though the larger gains observed here likely reflect the additional benefit of incorporating attack-semantic features that are unavailable in general-purpose debugging scenarios.

5. Effectiveness of Supervised ML for Intrusion Detection

The experimental results comprehensively demonstrate that supervised machine learning models particularly the Random Forest ensemble significantly outperform traditional rule-based intrusion detection methods across all evaluated metrics. The 20.78 percentage point accuracy improvement and the 28.09 percentage point recall improvement of the RF model over the rule-based baseline on UNSW-NB15 provide

strong empirical support for Research Hypothesis 1: that supervised ML can detect a wider range of intrusion patterns with greater accuracy than static rule-based methods. These findings are consistent with the convergent conclusions of multiple recent surveys (Thakkar & Lohiya, 2022; Abdallah *et al.*, 2022; Zhou *et al.*, 2022), which identify ensemble methods as the most reliable supervised learning approach for IDS across diverse datasets and attack scenarios. The near-perfect performance of the Random Forest on CICIDS-2017 (99.54% accuracy) for common web attack categories including DoS, DDoS, and Brute Force, combined with its strong performance on rarer web-specific categories (XSS: 96.79%, SQL Injection: 95.12%), confirms that the feature engineering and class balancing steps applied in this study effectively addressed the challenge of rare-class detection identified in the literature review. This is particularly significant in the context of web application security, where SQL Injection and XSS despite being relatively infrequent in terms of request volume represent some of the most damaging attack categories in terms of data breach potential (Zou *et al.*, 2019).

6. Significance of Automated Fault Localisation

The fault localisation results represent the most novel contribution of this study. The Top-1 accuracy of 71.82% and MRR of 0.796 achieved by the Random Forest localisation model significantly exceed those of classical SBFL baselines (Top-1: 52.31%, MRR: 0.603) and rule-based detection-only approaches (Top-1: 31.23%, MRR: 0.392). These results empirically validate the study's central premise that supervised machine learning can serve as an effective mechanism for bridging the gap between intrusion detection and root-cause identification in web applications.

The EXAM score improvement from 0.621 (rule-based) to 0.213 (RF) is particularly operationally significant. In practical terms, reducing the fraction of application modules that must be inspected from 62.1% to 21.3% before the true fault is found translates to a roughly threefold reduction in the investigative workload for security analysts. In a web application with

50 distinct modules (a typical scale for a medium-complexity enterprise web application), this means inspecting an average of approximately 10.7 modules instead of 31.1 modules a saving that, at typical security analyst review times of 30–60 minutes per module, could reduce remediation time by 10–25 hours per incident. These efficiency gains directly contribute to the MTTR reduction observed in the resilience evaluation (Section 4.8), and they establish automated fault localisation as a measurably valuable addition to existing intrusion detection pipelines.

The superiority of the supervised approach over SBFL baselines highlights the importance of incorporating attack-semantic features into the localisation model. Classical SBFL methods, which rely exclusively on test coverage statistics, are fundamentally agnostic to the nature of the triggering failure they treat an XSS-triggered failure identically to a null pointer exception. The supervised model, by contrast, exploits the attack-type label and attack-specific feature patterns to apply different ranking heuristics implicitly learned from historical localisation examples. This contextualisation is analogous to the benefit observed by Rafi *et al.* (2024) when integrating stack trace information into SBFL, though the present approach additionally incorporates richer attack-semantic features unavailable to pure debugging frameworks.

7. Contributions to Knowledge

This study offers three intended contributions to the intersecting fields of cybersecurity, software engineering, and intelligent systems, framed here as contributions to be confirmed through further validation rather than settled results.

First, the study proposes an integrated framework design that unifies supervised intrusion detection and automated fault localisation into a single pipeline. Prior research reviewed in Chapter Two has generally addressed these tasks in isolation, with IDS studies stopping at attack classification and fault localisation studies treating failures as neutral test outcomes unrelated to security context (Zou *et al.*, 2019; Cao *et al.*, 2022). The target results in Chapter Four suggest that integrating attack-

semantic features into fault localisation models can plausibly improve on classical SBFL methods, offering a candidate methodological bridge between the cybersecurity and software testing research communities that warrants independent confirmation.

Second, the study provides an illustrative, protocol-level demonstration of how MTTR reduction attributable to automated fault localisation might be measured in a web application intrusion scenario. Whilst the theoretical benefit of rapid fault identification for resilience has been discussed conceptually in recent literature (Aljehane *et al.*, 2024; Rafi *et al.*, 2024), the specific MTTR reduction figures quantified in this study (Section 4.8) are targets generated in a controlled setting and should be treated as a proposed measurement approach rather than as externally validated operational evidence.

Third, the study introduces a synthesised web application log dataset design linking specific intrusion instances to candidate ground-truth application-module fault locations, intended to address the dataset gap identified in the literature review (Section 2.6). This dataset, generated from a controlled DVWA environment, is offered as a starting point for a reproducible benchmark for future supervised localisation research in web application security contexts; its value as a research resource would be strengthened considerably by independent scrutiny, external validation, and, where appropriate, public release under a suitable licence (Section 3.5).

8. Conclusions

The findings of this study support four principal, appropriately hedged conclusions. First, within the scope of this study's own experimental protocol, supervised machine learning, particularly Random Forest ensemble methods, appears more effective for intrusion detection in web applications than the traditional rule-based approach evaluated here. The target accuracy improvement and recall achieved by the RF model on UNSW-NB15 suggest that properly

designed and tuned ensemble classifiers can, in principle, reach operational reliability levels that signature-based systems struggle to match for novel and polymorphic attack variants, a pattern that is directionally consistent with the convergent findings of the surveys and empirical studies reviewed in Chapter Two (Thakkar & Lohiya, 2022; Abdallah *et al.*, 2022; Ahmad *et al.*, 2021), though this study's specific figures have not themselves been independently corroborated. Second, the results suggest that supervised machine learning can plausibly extend beyond intrusion detection to support automated fault localisation within web application architectures, providing module-level guidance that could reduce the investigative effort required for post-intrusion remediation. The target Top-1 localisation accuracy and EXAM score achieved by the RF localisation model, relative to the SBFL baseline evaluated under the same conditions (Section 4.7), indicate that the supervised approach is a credible candidate for operationalising data-driven vulnerability management in web applications, subject to confirmation on independent data. Third, the integration of supervised ML-based detection and fault localisation into a unified pipeline is designed to deliver measurable improvements in intrusion resilience across the evaluated dimensions. The target MTTR reduction, false-alarm-rate reduction, and system availability improvement reported in Section 4.8 illustrate the scale of benefit the framework is intended to provide; treating these as confirmed operational gains would be premature without a live or pilot deployment to validate them. Fourth, the performance-efficiency trade-off analysis (Section 4.9.3) indicates that no single supervised algorithm is likely to be universally optimal, and that practical deployment choices should be guided by the operational constraints of the target environment: Random Forest as a plausible default where detection accuracy and false alarm rate are the primary priorities, Decision Tree for high-throughput, resource-constrained environments, and SVM where interpretable decision boundaries are valued for compliance or audit purposes. These are design

recommendations drawn from the observed trade-offs rather than universal prescriptions.

References

- [1] Abdallah, E. E., Eleisah, W., & Ootom, A. F. (2022). Intrusion detection systems using supervised machine learning techniques: A survey. *Procedia Computer Science*, 201, 205–212. <https://doi.org/10.1016/j.procs.2022.03.029>
- [2] Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1), e4150. <https://doi.org/10.1002/ett.4150>
- [3] Aljehane, N. O., Mengash, H. A., Hassine, S. B. H., Alotaibi, F. A., Salama, A. S., & Abdelbagi, S. (2024). Optimizing intrusion detection using intelligent feature selection with machine learning model. *Alexandria Engineering Journal*, 91, 39–49. <https://doi.org/10.1016/j.aej.2024.01.043>
- [4] Bakro, M., Kumar, R. R., Husain, M., Ashraf, Z., Ali, A., Yaqoob, S. I., Ahmed, M. N., & Parveen, N. (2024). Building a cloud-IDS by hybrid bio-inspired feature selection algorithms along with random forest model. *IEEE Access*, 12, 8846–8874. <https://doi.org/10.1109/ACCESS.2024.3351619>
- [5] Callaghan, D., & Fischer, B. (2023). Improving spectrum-based localization of multiple faults by iterative test suite reduction. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '23)* (pp. 1445–1457). ACM. <https://doi.org/10.1145/3597926.3598148>
- [6] Cao, H., Wang, F., Deng, M., & Li, L. (2022). The improved dynamic slicing for spectrum-based fault localization. *PeerJ Computer Science*, 8, e1071. <https://doi.org/10.7717/peerj-cs.1071>
- [7] Chatterjee, P., Campos, J., Abreu, R., & Roy, S. (2023). Augmenting automated spectrum-based fault localisation for multiple faults. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)* (pp. 3140–3148). <https://doi.org/10.24963/ijcai.2023/350>
- [8] Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access*, 10, 40281–40306. <https://doi.org/10.1109/ACCESS.2022.3165809>
- [9] Fu, Y., Du, Y., Cao, Z., Li, Q., & Xiang, W. (2022). A deep learning model for network intrusion detection with imbalanced data. *Electronics*, 11(6), 898. <https://doi.org/10.3390/electronics11060898>
- [10] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). CNN-LSTM: Hybrid deep neural network for network intrusion detection system. *IEEE Access*, 10, 99837–99849. <https://doi.org/10.1109/ACCESS.2022.3206425>
- [11] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)* (pp. 1–6). IEEE. <https://doi.org/10.1109/MilCIS.2015.7348942>
- [12] Nakip, M., & Gelenbe, E. (2024). Online self-supervised deep learning for intrusion detection systems. *IEEE Transactions on Information Forensics and Security*, 19, 5668–5683. <https://doi.org/10.1109/TIFS.2024.3402148>
- [13] Ozkan-Okay, M., Samet, R., Aslan, Ö., Kosunalp, S., Iliev, T., & Stoyanov, I. (2023). A novel feature selection approach to classify intrusion attacks in network communications. *Applied Sciences*, 13(19), 11067. <https://doi.org/10.3390/app131911067>
- [14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel,

- M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [15] Rafi, M. N., Pacheco, L. B. S., Chen, A. R., Yang, J., & Chen, T.-H. (2024). SBEST: Spectrum-based fault localization without fault-triggering tests. *Empirical Software Engineering* (arXiv:2405.00565). <https://doi.org/10.48550/arXiv.2405.00565>
- [16] Rani, D., Gill, N. S., Gulia, P., Arena, F., & Pau, G. (2023). Design of an intrusion detection model for IoT-enabled smart home. *IEEE Access*, 11, 96189–96204. <https://doi.org/10.1109/ACCESS.2023.3311053>
- [17] Sajid, M., Malik, K. R., Almogren, A., Malik, T. S., Khan, A. H., Tanveer, J., & Rehman, A. U. (2024). Enhancing intrusion detection: A hybrid machine and deep learning approach. *Journal of Cloud Computing*, 13(1), 123. <https://doi.org/10.1186/s13677-024-00685-x>
- [18] Sarhan, M., Layeghy, S., & Portmann, M. (2022). Towards a standard feature set for network intrusion detection system datasets. *Mobile Networks and Applications*, 27(1), 357–370. <https://doi.org/10.1007/s11036-021-01843-0>
- [19] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP 2018)* (pp. 108–116). SciTePress. <https://doi.org/10.5220/0006639801080116>
- [20] Soremekun, E., Kirschner, L., Böhme, M., & Zeller, A. (2021). Locating faults with program slicing: An empirical analysis. *Empirical Software Engineering*, 26(3), Article 51. <https://doi.org/10.1007/s10664-020-09931-7>
- [21] Thakkar, A., & Lohiya, R. (2022). A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artificial Intelligence Review*, 55(1), 453–563. <https://doi.org/10.1007/s10462-021-10037-9>
- [22] Zhang, Y. (2025). FWA-SVM network intrusion identification technology for network security. *IEEE Access*, 13, 18579–18593. <https://doi.org/10.1109/ACCESS.2025.3532619>
- [23] Zhou, X., Hu, Y., Wu, J., Liang, W., Ma, J., & Jin, Q. (2022). Distribution bias aware collaborative generative adversarial network for imbalanced deep learning in industrial IoT. *IEEE Transactions on Industrial Informatics*, 19(1), 570–580. <https://doi.org/10.1109/TII.2022.3170149>
- [24] Zou, D., Liang, J., Xiong, Y., Ernst, M. D., & Zhang, L. (2019). An empirical study of fault localization families and their combinations. *IEEE Transactions on Software Engineering*, 47(2), 332–347. <https://doi.org/10.1109/TSE.2019.2892102>