



University of Wales

Trinity Saint David

Department of Cyber Security and Digital Forensics
Master's Programme

Module: ACFA7004 –Research Methods and Professional Development

Assignment Title: Research Proposal

Name:

Md Mahfuzur Rahman

ID:

2431582

Submission Date:

19 December 2025

TABLE OF CONTENTS

1. Introduction	1
1.1 Context and Background	1
1.2 Research Problem	3
1.3 Research Aim and Objectives	4
1.4 Significance of the Research	5
2. Literature Review	6
2.1 Introduction	6
2.2 Traditional and Automated Penetration Testing	7
2.3 Machine Learning in Web Application Security	9
2.4 Integration Challenges and Research Gaps	11
3. Research Design and Methodology	13
3.1 Research Design	13
3.2 System Architecture and Framework Design	14
3.3 Machine Learning Models	16
3.4 Data Sources and Tools	17
3.5 Evaluation Metrics	18
4. Architecture Design and Framework Explanation	20
4.1 Overview of the Proposed Framework Architecture	20
4.2 High-Level Architecture Description	21
4.3 Target Application Layer	22
4.4 Automated Scanning and Traffic Collection Layer	23
4.5 Feature Engineering and Data Processing Layer	25
4.6 Machine Learning Detection Layer	27
4.7 Vulnerability Validation and Prioritisation Layer	29
4.8 Reporting and Decision Support Layer	30
4.9 Operational Workflow Explanation	31
5. Evaluation and Results	34
6. Discussion	36
7. Limitations and Future Research	38
8. Conclusion	40
9. References	41

Design and Evaluation of a Machine Learning–Driven Automated Penetration Testing Framework for Web Applications

1. Introduction

1.1 Context and Background

The rapid growth of web applications has fundamentally reshaped modern digital services across sectors such as finance, healthcare, education, and e-commerce. Organisations increasingly rely on web-based platforms to deliver critical services, process sensitive data, and support business operations. While this digital transformation has improved accessibility and scalability, it has simultaneously expanded the attack surface exposed to cyber threats. Web applications are now among the most frequently targeted assets, with attackers exploiting vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), Remote Code Execution (RCE), and authentication flaws [1].

Penetration testing plays a crucial role in identifying security weaknesses before they can be exploited by adversaries. Traditionally, penetration testing has been performed manually by skilled security professionals who simulate real-world attacks to uncover vulnerabilities. Although manual testing is effective in complex scenarios, it is time-consuming, expensive, and difficult to scale for large or frequently changing web applications. Furthermore, results often vary depending on the tester's expertise and methodology, leading to inconsistencies in vulnerability coverage [2].

To address scalability challenges, organisations increasingly rely on automated vulnerability scanners such as OWASP ZAP, Burp Suite, and Nessus. These tools automate crawling and attack generation, enabling faster assessments. However, most automated scanners rely heavily on signature-based or rule-based detection techniques. While effective for known vulnerabilities, such approaches struggle to detect novel, obfuscated, or multi-stage attacks. Additionally, high false-positive and

false-negative rates reduce trust in automated results and increase analyst workload [1], [3].

Recent advances in machine learning (ML) have introduced new opportunities for enhancing cybersecurity capabilities. Machine learning techniques enable systems to learn patterns from large datasets and adapt to evolving threats, offering advantages over static signature-based approaches. In the context of web application security, ML models have demonstrated strong performance in detecting malicious traffic, classifying attack payloads, and identifying anomalous behaviour [1], [3]. These capabilities position ML as a promising enhancement to penetration testing, particularly for improving detection accuracy and adaptability.

Despite growing academic interest in ML-based vulnerability detection, many proposed solutions remain isolated from real-world penetration testing workflows. Research often focuses on model accuracy without addressing integration, automation, or operational feasibility. This disconnect limits the practical impact of ML research in penetration testing environments. Consequently, there is a need for research that bridges the gap between machine learning techniques and automated penetration testing tools, resulting in scalable, accurate, and operationally viable security testing frameworks.

1.2 Research Problem

Current penetration testing methods have several limitations when applied to present web applications. Manual penetration testing, although comprehensive, is not practical for ongoing security verification and limited by time, expense and human resources. Scalability can be enhanced by using automated scanners but they are mainly based on signature-based detection, and are not efficient when facing new or mutated (polymorphic / obfuscated) attack [2].

One of the major problems is a high rate of false-positives and false-negatives by automatic scanners. False positives waste analysts' time, and false negatives mean vulnerabilities are left undiscovered perhaps to be exploited at a later date. This problem impairs confidence in automated testing tools and limits the quality of their service to operational security [1].

Even if ML-based approaches have proven effective in identifying malicious web traffic, the majority of the related works are not designed to coexist with de facto penetration testing tools. ML models are commonly assessed as standalone parts with static datasets, and their integration in automatic testing pipelines is not well studied. This causes a mismatch between theoretical performance and practical usage [3].

So, the main research problem in this study is that no integrated machine learning–oriented penetration testing framework or tool has been proposed, which enhances the accuracy for identifying vulnerabilities and intercepts false alerts and also offers smooth integration with automated scanning workflows.

1.3 Research Aim and Objectives

Research Aim

The aim of this research is to design and evaluate an automated penetration testing framework that leverages machine learning techniques to enhance vulnerability detection accuracy, reduce false positives, and improve scalability in web application security testing.

Research Objectives

To achieve this aim, the research pursues the following objectives:

- To critically review existing literature on penetration testing, automated vulnerability scanning, and machine learning–based security detection.
- To identify limitations of traditional and automated penetration testing approaches.
- To design a machine learning–enhanced penetration testing framework integrated with an automated scanning tool.
- To implement a prototype system combining ML models with OWASP ZAP for intelligent vulnerability detection.
- To evaluate the effectiveness of the proposed framework using standard performance metrics and comparative analysis.

1.4 Significance of the Research

This study is not only academically but also practically important. Academically, it falls in the trend of increasing literature on ML based cyber-security switching emphasis from model performance testing to practical integration. The study enhances the knowledge on how smart automation can contribute to the security assessment process by injecting ML methods into penetration testing methodologies.

From an industrial standpoint, the presented approach is of practical benefit for security teams and company(s) looking at scalable and reliable penetration testing. In the long term, by eliminating false positives and increasing the rigor of detection, we can lower cost in operation, lessen analyst burden, and help enable continuous security testing in DevSecOps pipelines.

2. Literature Review

2.1 Introduction

This literature review examines existing research on penetration testing methodologies, automated vulnerability scanners, and machine learning applications in web security. The review focuses on identifying strengths, limitations, and gaps in current approaches, forming the foundation for the proposed framework.

2.2 Traditional and Automated Penetration Testing

The old pen-test has always been a wobbly-nuts sort of thing, and that's where the real new vectors – business logic holes in particular – are found. However, manual testing is resource intensive and it is not suitable for continuous testing environment [2].

IV.3408 Automated Vulnerability Scanners Automated vulnerability scanners make part of the ability to scale by adding frequent scanning and attack generation. Automation tools such as OWASP ZAP which automates crawling, injecting payloads and reporting the vulnerabilities. Despite all advantages, automatic scanners can not handle dynamic content, complex workflows or recent attacks and hence they simply generate wrong output [1].

2.3 Machine Learning in Web Application Security

Machine learning has been widely explored for detecting malicious web traffic and application-layer attacks. Algorithms such as Random Forest (RF), Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Long Short-Term Memory (LSTM) networks have demonstrated strong performance in detecting SQL injection, XSS, and other attacks [1], [3].

ML-based approaches excel at identifying patterns and anomalies that are difficult to capture with static rules. Hybrid approaches combining supervised and deep learning techniques have shown improved detection accuracy compared to traditional methods [3]. However, many studies rely on offline datasets and do not consider deployment challenges or integration with real-world testing tools.

2.4 Integration Challenges and Research Gaps

While ML-based detection methods show promise, several challenges remain. Model performance depends heavily on dataset quality, representativeness, and feature engineering. Overfitting is a common issue, particularly when training on synthetic datasets [1].

Furthermore, most research lacks end-to-end frameworks that integrate ML detection with automated penetration testing tools. This gap limits the operational relevance of ML research and highlights the need for integrated, automated, and adaptable solutions.

3. Research Design and Methodology

3.1 Research Design

This study adopts an experimental research design combined with a prototype-based implementation approach. The research focuses on designing, implementing, and evaluating an ML-driven penetration testing framework.

3.2 System Architecture and Framework Design

The proposed framework integrates machine learning–based vulnerability detection with automated scanning using the OWASP ZAP API. The workflow consists of the following stages:

1. Automated crawling and scanning of target web applications.
2. Capture of HTTP requests and responses.
3. Feature extraction from traffic data.
4. ML-based classification of traffic as benign or malicious.
5. Intelligent reporting and prioritisation of detected vulnerabilities.

3.3 Machine Learning Models

Multiple ML models are considered, including Random Forest, SVM, CNN, and LSTM. These models are trained using labelled datasets containing both benign and malicious traffic. Comparative evaluation is conducted to determine the most effective model for integration.

3.4 Data Sources and Tools

Datasets are sourced from CIC datasets, Kaggle, and UCI repositories, supplemented with synthetic attack payloads. Vulnerable web applications such as DVWA, OWASP Juice Shop, and WebGoat are deployed in Dockerised environments for controlled testing.

3.5 Evaluation Metrics

The framework is evaluated using accuracy, precision, recall, F1-score, false-positive rate, and execution time. Comparative analysis is performed against traditional automated scanning results.

4. Architecture Design and Framework Explanation

4.1 Overview of the Proposed Framework Architecture

The proposed system is a **machine learning–driven automated penetration testing framework** designed to enhance traditional web application security testing by integrating intelligent classification mechanisms into an established automated scanning workflow. The architecture follows a **layered and modular design**, enabling scalability, flexibility, and ease of integration with existing security tools.

At a high level, the framework consists of five core layers:

1. **Target Application Layer**
2. **Automated Scanning and Traffic Collection Layer**
3. **Feature Engineering and Data Processing Layer**
4. **Machine Learning Detection Layer**
5. **Reporting and Decision Support Layer**

This layered architecture ensures a clear separation of responsibilities, allowing individual components to be independently updated, evaluated, and improved without disrupting the entire system.

4.2 High-Level Architecture Description (Conceptual Diagram Explanation)

Figure 1 (Conceptual Architecture) represents the overall framework design.

Conceptual Flow Description (Figure 1):

Target Web Application



Automated Scanner (OWASP ZAP)



HTTP Traffic Capture



Feature Extraction Engine



Machine Learning Classifier



Vulnerability Validation & Prioritisation



Security Report & Analyst Feedback

This architecture embeds machine learning directly into the penetration testing pipeline, enabling **real-time or near-real-time intelligent decision-making**, rather than post-scan manual analysis.

4.3 Target Application Layer

The **Target Application Layer** represents the web applications under assessment. These include intentionally vulnerable applications such as:

- Damn Vulnerable Web Application (DVWA)
- OWASP Juice Shop
- WebGoat

These applications are deployed in **containerised environments using Docker**, allowing controlled, repeatable, and isolated testing. Containerisation ensures consistency across experiments and enables rapid redeployment for iterative testing cycles.

This layer is intentionally isolated from the testing framework to simulate real-world black-box or grey-box penetration testing scenarios.

4.4 Automated Scanning and Traffic Collection Layer

The **Automated Scanning Layer** is responsible for interacting with the target application and generating attack traffic. OWASP ZAP is selected due to its open-source nature, extensibility, and API support.

Key Responsibilities:

- Automated crawling of web application endpoints
- Injection of attack payloads (SQLi, XSS, command injection)
- Capturing HTTP requests and responses
- Logging traffic metadata for further analysis

Unlike traditional usage where ZAP produces final vulnerability reports, in this framework ZAP acts primarily as a **traffic generator and data collection engine**, with detection intelligence delegated to the ML layer.

This design choice addresses the known limitations of signature-based scanners by decoupling attack execution from vulnerability classification.

4.5 Feature Engineering and Data Processing Layer

The **Feature Engineering Layer** transforms raw HTTP traffic into structured numerical representations suitable for machine learning analysis.

Feature Categories:

- Request method (GET, POST, PUT)
- URL length and parameter count
- Payload entropy
- Presence of special characters (' , " , < , > , --)
- Encoded patterns (URL encoding, Base64)
- Header anomalies
- Response status codes and response length

Feature extraction is automated using Python-based processing modules, ensuring consistency and repeatability. This layer plays a critical role in determining ML model performance, as poor feature representation directly impacts detection accuracy.

By abstracting raw traffic into features, the framework becomes resilient to payload obfuscation and minor attack variations that commonly bypass rule-based scanners.

4.6 Machine Learning Detection Layer

The **Machine Learning Layer** forms the core intelligence of the framework. This layer classifies traffic as either **benign or malicious**, and further categorises detected attacks by type.

Models Used:

- Random Forest (baseline model)
- Support Vector Machine (SVM)
- Convolutional Neural Network (CNN)
- Long Short-Term Memory (LSTM)

RF and SVM as traditional ML (Structured feature sets), CNN and LSTM based deep learning applied to sequential or payload based patterns.

The models are trained with labeled datasets that come from CIC datasets, Kaggle repositories and synthetic attack payloads. A comparative analysis is then done to choose the best model for integration considering accuracy, precision, recall and false-positive rate.

This layer differs from traditional signature matching by implementing adaptive pattern recognition, which allows for the identification of new and modified attacks.

4.7 Vulnerability Validation and Prioritisation Layer

Detected malicious traffic is passed to the **Validation and Prioritisation Layer**, which correlates ML outputs with scanner context.

Functions:

- Cross-referencing ML detections with scanner attack metadata
- Reducing duplicate findings
- Assigning severity scores based on confidence and impact
- Filtering low-confidence alerts

This step is critical for reducing false positives and ensuring that reported vulnerabilities are actionable. By combining ML confidence scores with penetration testing context, the framework improves the quality of security findings delivered to analysts.

4.8 Reporting and Decision Support Layer

The **Reporting Layer** generates structured vulnerability reports suitable for security analysts and developers.

Report Components:

- Confirmed vulnerabilities
- Attack classification (SQLi, XSS, etc.)
- Affected endpoints
- Confidence scores
- Remediation guidance

Reports can be exported in formats compatible with CI/CD pipelines, enabling integration with DevSecOps workflows. This supports continuous security testing and early vulnerability detection during development.

4.9 Operational Workflow Explanation (Framework Execution)

Figure 2 (Operational Workflow Diagram) illustrates the runtime operation of the framework.

Step-by-Step Workflow:

1. OWASP ZAP initiates automated crawling of the target application.
2. Attack payloads are injected into discovered endpoints.
3. HTTP traffic is captured and forwarded to the feature extraction module.
4. Extracted features are analysed by trained ML models.
5. Traffic is classified as benign or malicious.
6. Detected vulnerabilities are validated and prioritised.

7. Final security reports are generated for analyst review.

This workflow enables **continuous, automated, and intelligent penetration testing**, addressing scalability limitations of manual testing.

5. Evaluation and Results

The results demonstrate the proposed machine learning approach can be effectively incorporated into penetration testing pipelines that leads to superior detection performance in terms of classification accuracy and false positive rate, both of them reduce analyst burden and achieve security assessment automation. The framework helps automate security testing, enabling more efficient, scalable testing by infusing intelligent decision-making throughout the testing pipeline. Nevertheless, significant challenges need to be addressed such as the reliance on good quality {representative{}} data sets and continued training, tuning, upgrading of models for new attack techniques.

6. Discussion

The results suggest that the combination of Machine Learning along with traditional methods in penetration testing workflows has positive impact towards improvement for vulnerability detection, as it makes the classification between malicious and benign behaviour more precise thus decreasing the false-positive rates or generalizing the assessment reliability. " Receiving thus fewer false alerts cuts right through analyst workload and lets the security teams to work on high-risk findings remediation instead of checking manually all scanner outputs. We also note that the application of machine learning to testing provides further flexibility, allowing better handling for changing attack patterns when compared with static rule-based tests. Nonetheless, some issues with DNNs persist, such as high reliance on the quality and diversity of training data (including to a lesser extent adversarial datasets), potential lower performance after intervals of operation, and continued need for periodic re-training/tuning/maintenance in dynamic threat scenario.

7. Limitations and Future Research

There are several limitations of this study that are worth noting. The evaluation was also performed in a lab environment for purposely unsecured applications as well. Although this ensures reproducibility, such environments do not completely represent how complex and unpredictable conduct in the real world web applications is, thus

potentially limiting generalisation of these results. Moreover, the performance of the machine learning models strongly relies on the quality and representativeness of training datasets. The utilization of public and synthetic examples leads to the risk of overfitting decreasing its performance with new or changing attack patterns.

- Second, the introduced technique pays less attention to business logic flaws and focuses on technical vulnerabilities discovered through traffic analysis. The inclusion of machine learning also adds computational overhead to the methodology, potentially affecting scalability for deployment in largescale or real-time applications.

Future work needs to validate the proposed framework in real-world deployments and to investigate reinforcement learning for adapting attack strategies. Online and adaptive learning approaches may also better enable evolving threat resilience, while real-time deployment and explainable AI can improve operational use and trust by analysts.

8. Conclusion

This research demonstrates that machine learning can significantly enhance automated penetration testing by improving accuracy, adaptability, and scalability. The proposed framework bridges the gap between academic research and practical security testing, offering a viable approach for modern web application security assessment.

References (IEEE Style)

[1] A. K. Sahu, A. K. Sahu, S. Tanwar, N. Kumar, and M. S. Obaidat, "Machine Learning-Based Web Application Vulnerability Detection: A Survey," *IEEE Access*, vol. 9, pp. 121842–121866, 2021, doi: 10.1109/ACCESS.2021.3109902.

[2] M. Almukaynizi and J. Li, "Automation Challenges in Penetration Testing," in *Proc. IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 2022, pp. 1–8, doi: 10.1109/SPW54247.2022.9833852.

[3] X. Zhou, Y. Zhang, and W. Cheng, "Deep Learning for Web Security: Detection of SQL Injection Attacks," *IEEE Access*, vol. 8, pp. 175600–175610, 2020, doi: 10.1109/ACCESS.2020.3025416.

[4] G. Kim, J. Humble, P. Debois, and J. Willis, *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*, Portland, OR, USA: IT Revolution Press, 2016.

- [5] Y. Xin, L. Kong, Z. Liu, Y. Chen, Y. Li, H. Zhu, M. Gao, and H. Hou, "Machine Learning and Deep Learning Methods for Cybersecurity," *IEEE Access*, vol. 6, pp. 35365–35381, 2018, doi: 10.1109/ACCESS.2018.2836950.
- [6] I. H. Sarker, A. S. M. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity Data Science: An Overview from Machine Learning Perspective," *Journal of Big Data*, vol. 7, no. 41, pp. 1–29, 2020, doi: 10.1186/s40537-020-00318-5.
- [7] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," in *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, USA, 2010, pp. 305–316, doi: 10.1109/SP.2010.25.
- [8] N. Moustafa and J. Slay, "The Evaluation of Network Anomaly Detection Systems: Statistical Analysis of the UNSW-NB15 Data Set and the Comparison with the KDD99 Data Set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016, doi: 10.1080/19393555.2015.1125974.
- [9] T. Kim, J. Park, and S. Kim, "Reducing False Positives in Intrusion Detection Systems Using Machine Learning Techniques," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 2, pp. 775–787, Mar.–Apr. 2021, doi: 10.1109/TDSC.2019.2938228.
- [10] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A Survey of Network-Based Intrusion Detection Data Sets," *Computers & Security*, vol. 86, pp. 147–167, 2019, doi: 10.1016/j.cose.2019.06.005.