

AUTOMATED WEB APPLICATION PENETRATION TESTING USING MACHINE LEARNING

Md Mahfuzur Rahman

MSc Cybersecurity & Digital Forensics

Date: 8.12.2025



Research Context

- Rapid growth of web applications has **expanded the attack surface**, increasing exposure to security risks [1], [2].
- Surge in sophisticated attacks such as **SQL Injection, XSS, and Remote Code Execution (RCE)** challenges traditional defences [1], [3].
- Manual penetration testing** is time-consuming, inconsistent across testers, and costly to scale for continuous security needs [2].
- Automated scanners** often suffer from limited accuracy, poor adaptability, and difficulty detecting complex or evolving attacks [1], [2].
- Machine learning** presents a promising solution for **intelligent, adaptive vulnerability detection**, outperforming signature-only methods in many scenarios [1], [3].

Research Problem

- Signature-based scanners fail to detect **novel, obfuscated, or multi-stage attacks**, limiting effectiveness in modern threat environments [2].
- High false-positive and false-negative rates** in traditional scanners reduce trust and create additional analyst workload [1], [2].
- ML vulnerability research is often **isolated and not integrated** into operational penetration testing workflows [1], [3].
- Research Gap:** Need for an **ML-driven automated penetration testing framework** that improves detection accuracy, reduces false alerts, and scales efficiently to complex web applications [1], [2], [3].

Research Aim & Objectives

- ML-Driven Detection:** Uses ML/deep learning models to identify web vulnerabilities with higher accuracy and reduced false positives [1], [3].
- Integrated Workflow:** Combines ML classifier outputs with automated scanning via OWASP ZAP API for intelligent, adaptive testing [2].
- Automated Data Pipeline:** Captures HTTP traffic, extracts features, and feeds results into ML models in real time.
- Prototype Implementation:** End-to-end system using trained ML models + ZAP automated scanning and reporting.
- Performance Evaluation:** Measures accuracy, precision, recall, F1-score, and false-positive rates to validate effectiveness [1].
- Benchmarking:** Compares ML-enhanced detection outcomes with traditional scanners to demonstrate improved coverage and efficiency [2], [3].

Literature Review Summary

- ML models such as RF, SVM, CNN, and LSTM demonstrate strong performance in detecting common web vulnerabilities [1], [3].
- Traditional automated scanners struggle with dynamic content, complex workflows, and evolving attack patterns [2].
- Hybrid ML approaches enhance anomaly detection and classification accuracy compared to rule-based tools [1], [3].
- **Research Gap:** Lack of ML-driven frameworks integrated into fully automated, real-world pentesting systems [2].

Research Methodology

- **Approach:** Experimental research with ML-based vulnerability detection [1], [3].
- **Data Sources:** CIC datasets, Kaggle, UCI repositories, plus synthetic attack payloads for broader coverage [1].
- **ML Models:** Random Forest, SVM, CNN/LSTM used for classifying malicious vs. benign traffic [3].
- **Workflow:** ZAP automated crawl → feature extraction → ML classification → reporting [2].
- **Evaluation:** Precision, Recall, F1-score, false positives, execution time [1]–[3].

Project Planning & Resources

- **Resources**
- Python ML frameworks (Scikit-learn, TensorFlow) for model training and evaluation [1], [3]
- OWASP ZAP API for automated scanning and data extraction [2]
- Dockerised vulnerable web apps (DVWA, Juice Shop, WebGoat) for controlled testing environments [1]
- Kali Linux for penetration-testing operations and workflow automation [2]
- **Project Timeline (14 Weeks)**
- **Weeks 1–2:** Literature Review
- **Weeks 3–4:** Dataset Preparation
- **Weeks 5–7:** ML Model Development
- **Weeks 8–10:** System Integration
- **Weeks 11–12:** Evaluation
- **Weeks 13–14:** Final Report & Presentation

Project Impact & Limitations

- **Impact**
- Improves vulnerability detection accuracy and testing speed [1], [3].
- Reduces false positives and analyst workload through ML-enhanced classification [1].
- Supports automated DevSecOps workflows and CI/CD security integration [2].
- Strengthens overall web application security posture by enabling earlier vulnerability identification [1]–[3].
- **Limitations**
- ML performance highly dependent on dataset quality and representativeness [1].
- Risk of overfitting when trained on synthetic or controlled environments [3].
- ML-based systems cannot fully detect business-logic or contextual vulnerabilities [2].
- Requires continuous retraining, tuning, and model maintenance to remain effective [1].

References

- [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.
- [2] M. Almkaynizi and J. Li,
– “Automation Challenges in Penetration Testing,”
– in IEEE Security & Privacy Workshops (SPW), 2022.
- [3] X. Zhou, Y. Zhang, and W. Cheng,
– “Deep Learning for Web Security: SQL Injection Detection,”
– IEEE Access, vol. 8, pp. 175600–175610, 2020.