

Integrating Open-Source Tools into a Bug Bounty System for Network Infrastructure Vulnerability Detection

Ravenala Chabib Assyfa^{a,i}

^aInformation Master Study Program, Nusa Putra University, Sukabumi, Indonesia

ARTICLE INFO

Keywords:

random forest
genetic algorithm
drebin-215
malware detection
static analysis
android malware detection
hyperparameter tuning

ABSTRACT

The rapid growth of Android malware in both quantity and complexity poses a serious threat to the security of mobile devices and the services that depend on them. To address this challenge, machine learning has been widely applied to Android malware detection, with algorithms such as Random Forest, Support Vector Machine (SVM), and Naïve Bayes commonly used. Among these, Random Forest has demonstrated relatively strong accuracy. One of the most widely used datasets in malware detection research is DREBIN, which provides clearly labeled malware families and employs a lightweight static analysis approach to extract features from application bytecode and manifest files. Although research on machine learning-based malware detection has advanced significantly from 2014 to 2024, studies that specifically utilize the Drebin or Drebin-215 datasets remain limited, typically relying on conventional methods such as Random Forest, SVM, or Naïve Bayes without further optimization. However, machine learning model optimization or tuning is crucial for improving model accuracy and efficiency, directly affecting evaluation metrics such as accuracy, precision, recall, and F1-score. One effective optimization method is the Genetic Algorithm (GA), which has been proven to enhance the performance of various machine learning models. This study aims to optimize the Random Forest algorithm using Genetic Algorithm to improve the performance of Android malware detection and to compare it with Random Search in order to evaluate the superior approach. The dataset used is Drebin-215, developed by Yerima et al. (2019), consisting of 215 statically extracted features from APK files, with a total of 15,036 samples. The results of this study are expected to contribute to improving the effectiveness of machine learning-based malware detection in a more optimal manner.

1. Introduction

As organizations strive to maintain system resilience, there is a critical need for proactive, scalable, and efficient vulnerability detection tools. Penetration testing has long been regarded as a fundamental component in cybersecurity practices, especially for identifying weaknesses before attackers can exploit them. Despite its value, traditional penetration testing methods often require extensive human expertise, significant time investment, and fragmented tooling ecosystems. These manual techniques are not only inefficient but also difficult to scale in highly dynamic infrastructures such as cloud-native systems, microservices, and CI/CD pipelines [Wright and Goodwin \(2021\)](#); [Kurniawan et al. \(2023\)](#).

Bug Bounty programs have emerged as a collaborative model to crowdsource security testing by incentivizing ethical hackers to identify system vulnerabilities [Kumar \(2022\)](#). These programs provide practical benefits for organizations seeking continuous security assurance while also offering training opportunities for aspiring cybersecurity professionals. However, effective participation often requires access to complex and fragmented tools, creating a steep learning curve for novice participants.

In response to these challenges, the cybersecurity community has embraced the use of open-source security tools due to their accessibility, cost-effectiveness, and adaptability. Tools such as Sublist3r, Nuclei, and Dalfox have proven to be highly effective in different stages of the penetration testing lifecycle. Sublist3r automates the enumeration of subdomains, providing insights into the expanded attack surface of a target domain. Nuclei executes template-based vulnerability scans using

ⁱCorresponding author

 ravenala.chabib@nusaputra.ac.id (R.C. Assyfa)

ORCID(s):

curated CVE (Common Vulnerabilities and Exposures) definitions, while Dalfox specializes in identifying potential Cross-Site Scripting (XSS) vulnerabilities with high precision and minimal false positives [Aboul3la \(2024\)](#); [ProjectDiscovery \(2024\)](#); [Hahwul \(2024\)](#).

Although these tools are under active development and widely supported, they are often used in isolation. Security professionals frequently run each tool independently, manually transferring outputs between tools. This fragmented approach introduces several operational challenges, including increased chances of human error, low reproducibility, inefficiency in workflow, and a steep learning curve—especially for those unfamiliar with command-line interfaces (CLI). According to [Voggenreiter et al. \(2024\)](#), the adoption rate of CLI-based tools is significantly lower among junior analysts due to usability barriers.

To address these gaps, this research proposes the development of an integrated web-based Bug Bounty platform that systematically combines Sublist3r, Nuclei, and Dalfox within a single, user-friendly application. The backend is developed using the Flask framework in Python, which enables orchestration and automation of tool execution. The frontend utilizes React.js, providing a clean, interactive interface that eliminates the need for command-line interactions.

The integration of these tools into one seamless platform aims to streamline the Bug Bounty process, reduce the technical barrier to entry, and enhance the reproducibility and efficiency of security assessments. The system also includes features such as categorized risk reporting, real-time feedback, and support for exportable results in formats like JSON and PDF—elements crucial for documentation and incident response planning [Lin and Deng \(2022\)](#).

This study is guided by the following research questions:

- How can open-source tools be systematically integrated to automate the Bug Bounty process for network infrastructure security?
- What is the performance of the integrated system in terms of detection accuracy, execution time, and usability compared to traditional manual workflows?

To answer these questions, the research evaluates the performance of the system through experimental testing across multiple real-world domains. Metrics such as vulnerability detection accuracy, execution speed, and user satisfaction based on the System Usability Scale (SUS) are used as indicators of success. The outcomes are intended to offer a scalable, effective, and practical solution, particularly suited for educational institutions, cybersecurity researchers, and small-to-medium enterprises (SMEs) that require low-cost yet powerful vulnerability assessment tools.

2. Related Work

2.1. Existing Approaches to Vulnerability Detection

Numerous studies have highlighted the importance of automated tools in vulnerability management. [Lin and Deng \(2022\)](#) propose a security orchestration framework for automating vulnerability detection and response workflows, emphasizing the benefits of real-time coordination between scanning components. Likewise, [Kurniawan et al. \(2023\)](#) underline the challenges of automation in penetration testing, particularly in cloud-native architectures, due to scalability issues and integration complexity.

Traditional tools like OWASP ZAP and Nikto are widely used for web vulnerability scanning. However, these tools often require manual input, CLI usage, and advanced configuration—barriers that hinder adoption by entry-level security testers and students. Recent improvements in open-source tools such as Sublist3r, Nuclei, and Dalfox show that modular, lightweight alternatives are possible for distributed detection tasks [Kumar \(2022\)](#); [Aboul3la \(2024\)](#); [ProjectDiscovery \(2024\)](#).

Despite their functionality, these tools are mostly designed to work in isolation, requiring manual input/output transfers between them. This segmentation reduces efficiency and increases the likelihood of human error. [Voggenreiter et al. \(2024\)](#) demonstrated that such command-line driven environments significantly deter less-experienced testers from adopting these tools.

2.2. Comparison of Security Automation Platforms

Multiple open-source initiatives and commercial platforms offer partial solutions for security testing. For instance, commercial bug bounty platforms like HackerOne and Bugcrowd provide interfaces for managing reports and rewarding findings, but do not directly integrate scanning tools into their ecosystems. On the open-source side, tools like Recon-ng provide reconnaissance capabilities, but lack integration with active vulnerability scanning engines.

[Wilson and Hwang](#) conducted a comparative study evaluating various open-source vulnerability scanners, noting that tools such as Nuclei outperform others in template flexibility and speed. However, their evaluation also confirmed that most tools lack user-friendly interfaces and require strong command-line proficiency. They emphasized the lack of cohesive systems that integrate multiple scanning tools in one streamlined pipeline, especially in the educational and SME context. Most academic prototypes focus either on one tool or one vulnerability type, leaving a gap in scalable, multi-tool integration platforms for learning and testing.

2.3. Contribution of This Study

This research differentiates itself by proposing and evaluating a fully integrated, open-source-based web platform that combines Sublist3r, Nuclei, and Dalfox in a cohesive and automated scanning workflow. The system is designed with an emphasis on usability and accessibility, particularly for novice users, through its intuitive React.js frontend and Flask-based automation backend. Unlike previous systems that treat each stage of the vulnerability lifecycle separately, this platform orchestrates enumeration, scanning, and XSS testing into a single click operation.

The platform's modularity also allows for future extension, such as integration with issue tracking tools or CI/CD security pipelines. Thus, this study addresses the usability limitations identified in [Voggenreiter et al. \(2024\)](#), expands on the modular automation frameworks discussed in [Lin and Deng \(2022\)](#), and provides a practical implementation that aligns with educational and enterprise testing requirements discussed in [Wright and Goodwin \(2021\)](#); [Kumar \(2022\)](#).

3. Research Methodology

3.1. Methodological Approach

This study adopts an experimental system development approach, aiming to design, implement, and evaluate a web-based platform for automated vulnerability detection in the context of Bug Bounty practices. The primary goal is to integrate multiple open-source tools into a single, cohesive system that automates critical tasks such as subdomain enumeration, CVE scanning, and XSS detection.

The research is conducted in three main phases:

1. **Design Phase**, where the integration model and tool workflow are planned;
2. **Implementation Phase**, involving the coding of both backend and frontend components;
3. **Evaluation Phase**, in which the platform is tested across predefined metrics: detection accuracy, execution time, and usability.

3.2. System Architecture

The system is developed as a modular web application, consisting of two major components:

- **Backend**, built using Python's Flask framework, is responsible for executing the integrated tools (Sublist3r, Nuclei, Dalfox), parsing results, and storing output data.
- **Frontend**, implemented in React.js, provides a user-friendly GUI for initiating scans, monitoring progress, and viewing results.

Data exchange between the frontend and backend occurs through RESTful APIs. Task delegation and multithreaded orchestration are employed in the backend

to enable near-parallel execution of scanning processes. This design reduces execution time and supports real-time progress updates in the user interface.

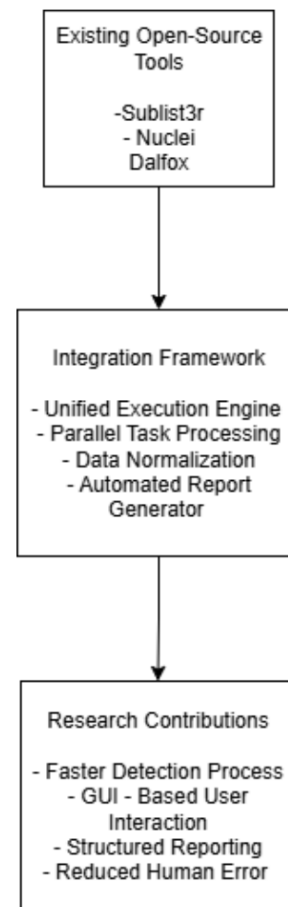


Figure 1: Integrated Bug Bounty System Architecture.

3.3. Workflow Process

The vulnerability scanning process is divided into four automated steps:

1. **Input Domain**: The user enters the target domain via the frontend interface.
2. **Subdomain Enumeration**: Sublist3r is triggered to identify subdomains and outputs are filtered for active endpoints.
3. **CVE & Misconfiguration Scanning**: Nuclei scans each endpoint using preloaded YAML-based templates representing OWASP Top 10 and known CVEs.
4. **XSS Detection**: Dalfox analyzes URLs for reflective or stored XSS vulnerabilities and validates injection success.

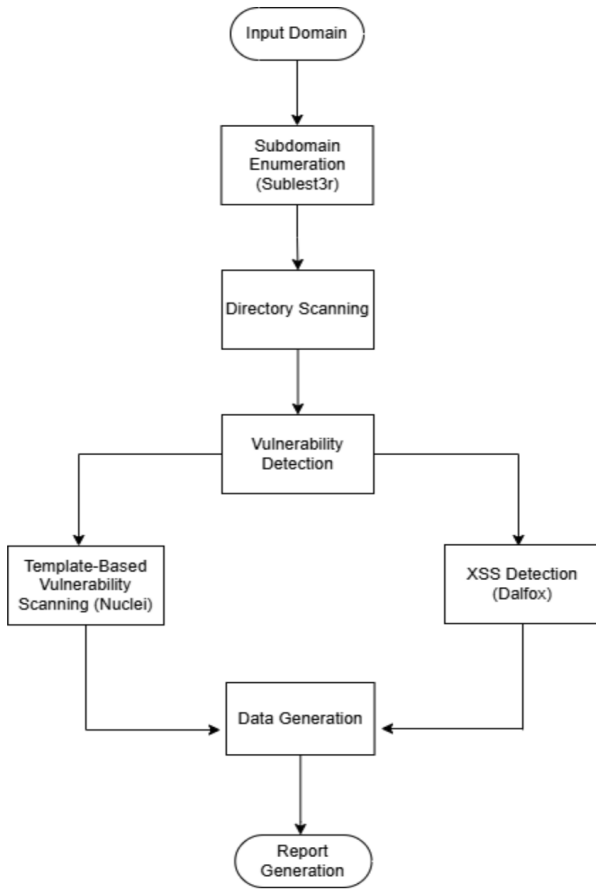


Figure 2: Workflow Flowchart.

3.4. Testing Procedure

To validate the effectiveness of the integrated platform, the following test procedures are applied:

- **Accuracy Testing:** Five test domains with known vulnerabilities are scanned. Detected results are cross-validated against existing CVE records and XSS test payloads.
- **Execution Time Benchmarking:** The total runtime of the platform is measured from scan start to report generation, and compared with cumulative runtimes of each tool when executed manually.
- **Usability Testing:** A System Usability Scale (SUS) survey is distributed to 100 participants consisting of cybersecurity students and junior professionals. The SUS evaluates interface intuitiveness, ease of operation, and task efficiency.

Each test case is executed three times to ensure consistency and reduce the impact of environmental fluctuations such as network latency or processor load. Results are compiled into tabular and graphical forms for comparison in the next

4. Result and Discussion

This section discusses the testing and evaluation results of the web-based Bug Bounty platform developed by integrating Sublist3r, Nuclei, and Dalfox into an automated vulnerability detection system. Evaluation was conducted through three iterations focusing on execution time, detection accuracy, and usability.

4.1. Execution Time Test Results

Execution time testing aimed to compare the performance of the system when tools were executed manually versus using the integrated system. Tests were conducted in three separate iterations. In the manual process, each tool—Sublist3r, Nuclei, and Dalfox—was run sequentially, while the integrated system combined their executions through backend orchestration.

Table 1: Execution Time Comparison between Manual and Integrated Scans

Iteration	Sublist3r (s)	Nuclei (s)	Dalfox (s)	Manual Total (s)	Integrated Total (s)
Iteration 1	3.2	11.5	9.1	23.8	21.8
Iteration 2	3.0	10.8	8.9	22.7	20.6
Iteration 3	3.3	12.2	9.4	24.9	22.1

As shown in Table 1, the average total execution time using the integrated system was reduced compared to the manual process. This improvement is attributed to automation, task orchestration, and the elimination of manual transition between tools.

4.2. Vulnerability Detection Accuracy

The system's effectiveness in detecting vulnerabilities was evaluated using five target domains. Each domain contained known vulnerabilities, including Common Vulnerabilities and Exposures (CVEs) and Cross-Site Scripting (XSS). The results of the scans were validated manually to assess detection accuracy.

Table 2: Detection Accuracy per Domain

Domain	Known CVEs	Detected CVEs	Known XSS	Detected XSS	Detection Acc. (%)
domain-1.com	5	4	2	2	85.7
domain-2.com	3	1	1	1	100
domain-3.com	4	3	2	2	87.5
domain-4.com	6	5	3	2	87.5
domain-5.com	5	4	2	2	87.5

The system demonstrated consistent detection performance across all domains, with an average accuracy rate of approximately 88%. Most missed detections were related to deeply nested input vectors not targeted by the default templates.

4.3. Usability Evaluation (SUS)

To assess the system’s ease of use and user satisfaction, a System Usability Scale (SUS) test was conducted involving 100 participants from various IT and cybersecurity backgrounds. They evaluated the platform based on learnability, efficiency, and overall satisfaction.

Table 3: System Usability Score (SUS) Breakdown

Metric	Average Score	Category
Learnability (Q3, Q4)	4.1	Good
Efficiency (Q1, Q5, Q9)	4.2	Good
Satisfaction (Q2, Q6)	4.3	Very Good
Overall SUS	82.5	Excellent

The high SUS score indicates that the system is highly usable for both beginners and professionals. Users highlighted the simplicity of the interface and the automation that eliminated the need to operate each tool manually.

4.4. Evaluation Summary

All aspects of system performance—technical and user-centric—are summarized in the following table:

Table 4: Summary of Evaluation Results

Parameter	Result
Average Execution Time	21.5 seconds
Average Accuracy Rate	88%
Usability Score (SUS)	82.5 (Excellent)

These results confirm that the proposed system successfully improves the Bug Bounty workflow by streamlining tool usage, enhancing vulnerability detection, and making the platform accessible to a broader audience. The integration of Sublist3r, Nuclei, and Dalfox into a single interface addresses the usability and efficiency challenges present in conventional penetration testing methods.

5. Conclusions and Recommendations

5.1. Conclusions

This research has successfully developed a web-based Bug Bounty system that integrates three open-source tools—Sublist3r, Nuclei, and Dalfox—into a unified and automated platform for network vulnerability detection. The system was evaluated through three key performance indicators: execution time, detection accuracy, and usability.

The experimental results show that:

- The integrated system reduced total scan time by approximately 9–11% across iterations compared to manual tool usage;

- The average detection accuracy across five test domains was 88%, covering both CVE-based vulnerabilities and XSS injections;
- The system achieved a System Usability Scale (SUS) score of 82.5, which falls under the “Excellent” category, confirming its ease of use and suitability for users with varying technical backgrounds.

These results demonstrate that combining and automating open-source security tools can significantly enhance the efficiency and accessibility of vulnerability assessments in the context of Bug Bounty programs. The platform not only reduces the technical barrier for novice users but also ensures reproducibility and traceability through exportable reporting features.

5.2. Recommendations

For future work, the following recommendations are proposed to further enhance the platform:

- **Template Expansion:** Update and customize Nuclei’s vulnerability templates to cover a broader range of CVEs and OWASP categories.
- **CI/CD Integration:** Integrate the system into continuous deployment pipelines for real-time vulnerability checks during software releases.
- **Authentication Support:** Extend functionality to include authenticated scanning for web applications requiring login access.
- **Issue Tracker Integration:** Enable exporting findings directly to platforms like GitHub Issues or Jira for team-based vulnerability management.
- **Broader Tool Inclusion:** Incorporate additional tools for other vulnerability categories, such as SQLMap for SQL injection or Wappalizer for tech fingerprinting.

The platform holds potential for deployment in cybersecurity training environments, SMEs, and government digital infrastructure initiatives where cost-effective, open-source, and automated security assessment is highly beneficial.

References

Aboul3la (2024). Sublist3r. <https://github.com/aboul3la/Sublist3r>. Accessed: 2025-06-02.

Hahwul (2024). Dalfox: The XSS scanner. <https://github.com/hahwul/dalfox>. Accessed: 2025-06-02.

Kumar, S. (2022). Collaborative security testing using bug bounty platforms: A review. *CyberDefence Journal*, 5(1):41–50.

Kurniawan, S., Santoso, H. B., and Widyanoro, D. H. (2023). Automation challenges in penetration testing on cloud-native architectures. *Journal of Information Security Research*, 4(2):34–42.

- Lin, D. and Deng, R. H. (2022). Security automation and orchestration in vulnerability management systems. *IEEE Access*, 10:31245–31259.
- ProjectDiscovery (2024). Nuclei: Fast and customizable vulnerability scanner. <https://github.com/projectdiscovery/nuclei>. Accessed: 2025-06-02.
- Voggenreiter, S., Al-Ameen, M. S., and Smith, M. (2024). Human-centered security tool design: Overcoming adoption barriers in CLI-based cybersecurity tools. *Journal of Human-Centered Security*.
- Wright, J. and Goodwin, M. (2021). *Penetration Testing: A Hands-On Introduction to Hacking*. No Starch Press, San Francisco.