PCMJ

Progress in Computer and Mathematics Journal (PCMJ)
volume 2 [August, 2025]
e-ISSN: 3030-6728
Website: fskmjebat.uitm.edu.my/pcmj

AI RECOMMENDATION PENETRATION TESTING TOOLS FOR PASSWORD ATTACKS: RANDOM FOREST

NURULASYIQIN ROZMAN
College of Computing Informatics and Mathematics, Universiti Teknologi MARA (UiTM), Jasin 77300 Melaka, Malaysia;
nurulasyiqinbintirozman@gmail.com

SHAHADAN SAAD*
College of Computing Informatics and Mathematics, Universiti Teknologi MARA (UiTM), Jasin 77300 Melaka, Malaysia;
shahadan@fskm.uitm.edu.my

Article Info

Abstract

This project developed an AI-based recommendation system using the Random Forest algorithm to address the complexities of selecting appropriate penetration testing tools for password attacks. Penetration testing, crucial for evaluating network and system security, faces challenges due to the variety of tools, especially for less experienced pentesters. The project's objective was to automate tool selection based on user-defined requirements, improving efficiency and effectiveness. Guided by the Extreme Programming (XP) methodology, the AI system analyzed attributes and requirements to provide personalized tool recommendations, such as Nmap, Medusa, Hydra, and Wfuzz, based on password attack types, targeted platforms, software types, hash types, and pentest goals. Implemented using Django and SQLite, the system reduced manual efforts and specialized knowledge needed for tool selection, allowing pentesters to focus on complex security tasks. The project's seamless integration with existing workflows demonstrated its practical capability and highlighted AI's potential in optimizing security practices, making pentesting more accessible for organizations with limited resources and expertise. By shifting focus from repetitive tasks to higher-level security analysis, the project enhanced organizational security against evolving cyber threats and showcased AI's role in improving cybersecurity practices.

Received: August 2024
Accepted: March 2025
Available Online: August 2025

Keywords: Pentesting; Extreme Programming; Django; SQLite

INTRODUCTION

Penetration testing emerged in the 1970s as organizations recognized the need to assess network and application security. Over time, it has become essential for identifying and fixing security vulnerabilities before they are exploited. However, the increasing variety of penetration testing tools, particularly for password attacks, has introduced challenges in selecting the right tool. Less experienced pentesters face difficulties in manually assessing and choosing appropriate tools, making the process time-consuming and error-prone, which can lead to incorrect tool selection and overlooked vulnerabilities.

The project aimed to develop an AI-based recommendation system using the Random Forest algorithm to automate the selection of penetration testing tools for password attacks. The specific objectives were to design a recommendation system that identifies suitable tools based on user requirements, enhance the efficiency of tool selection for tools like Nmap, Medusa, Hydra, and Wfuzz, and evaluate the model through unit and integration testing. The project utilized Django as the framework and SQLite as the database, focusing on providing personalized tool recommendations based on attributes such as attack type, target platform, and hash type.

The AI recommendation system significantly improved penetration testing by reducing the manual effort and specialized knowledge required, making it more accessible to organizations with limited resources. The automated selection process enhanced efficiency and effectiveness by quickly processing large amounts of data to identify vulnerabilities and recommend tools in real-time. Seamless integration with existing workflows allowed organizations to adopt the system without major changes. Overall, the project demonstrated AI's potential in optimizing security practices, shifting the focus from repetitive tasks to higher-level analysis and risk assessment, thus enhancing organizational security and business relationships.

LITERATURE REVIEW

Dalwani (2023) highlights the importance of penetration testing, an ancient method used by the Department of Defence, in assessing computer system security and preventing security vulnerabilities in information systems and services, reflecting the constant technological evolution. A group of professionals is developing an automated tool that combines the expertise of skilled penetration testers, allowing non-expert users to replace the penetration team. This approach offers an inclusive perspective on an organization's system security, reducing the time and cost of manual testing (Saber et al.,2023). Automated penetration testing offers standardized processes, updated attack databases, pre-built exploits, automated cleanup, consistent reporting, non-intrusive testing, detailed auditing, and simpler training requirements.

Random Forest Algorithm

The Random Forest algorithm is a supervised machine learning technique used in classification and regression tasks by constructing multiple decision trees during training to improve predictive accuracy and reduce overfitting (Zhang et al., 2021). It employs bootstrap sampling, where random subsets of training data are generated with replacement, and predictions are made by averaging the outputs for regression or taking the majority vote for classification (Lifandali et al., 2023).

The algorithm estimates feature importance by measuring the total decrease in node impurity and is widely used in finance, healthcare, and e-commerce (Zhang et al., 2021). Its ensemble approach, involving multiple decision trees, mitigates the risk of overfitting and enhances accuracy and robustness, making it suitable for developing an AI recommendation tool for password attack penetration testing by handling diverse input features and large datasets (Schonlau & Zou, 2020).

Extreme Programming (XP)

Extreme Programming (XP) is an agile methodology introduced by Kent Beck in the late 1990s to enhance flexibility and responsiveness in software development (Yasvi et al., 2019). XP's iterative and collaborative approach encourages continuous code refinement based on feedback, which helps eliminate the fear of making changes. This method actively involves customers to prioritize their needs, making it suitable for dynamic environments with frequently changing requirements.

Progress in Computer and Mathematics Journal (PCMJ)
volume 2 [August, 2025]
e-ISSN: 3030-6728
Website: fskmjebat.uitm.edu.my/pcmj

XP emphasizes rigorous testing, continuous integration, and simplicity, which collectively improve code maintainability and reliability (Yasvi et al., 2019).

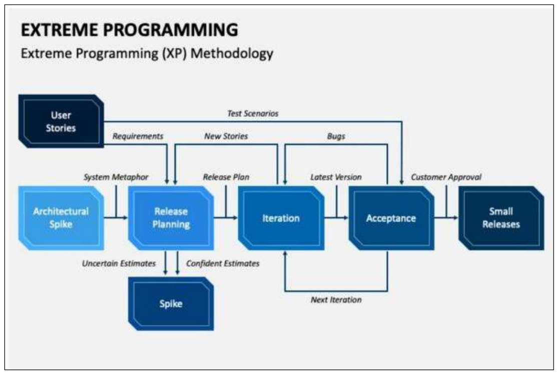


Figure 1.0 XP Methodology
(Source: Meier, 2023)

In the planning stage, user stories are identified, ranked, and estimated to create a customer-focused approach. This involves frequent small releases for quick user feedback and a collaborative effort to prioritize high-value features early in development (Shrivastava et al., 2021). The managing stage focuses on iterative development with small releases and continuous progress monitoring, maintaining flexibility to adapt to changing requirements, and ensuring a sustainable delivery pace (Yasvi et al., 2019). The designing stage promotes simplicity and adaptability, focusing on ease of understanding and modification to meet changing needs. This involves using design tools like CRC cards, "spike solutions" for technical challenges, and regular refactoring to enhance maintainability (Yasvi et al., 2019; Shrivastava et al., 2021). In the coding stage, the goal is to produce clean, efficient code that meets user story requirements, with an emphasis on continuous integration and test-driven development to ensure code quality and efficiency (Yasvi et al., 2019). The testing stage involves continuous automated testing, with tests written before the code to ensure reliability and early error detection. All code must pass unit tests before release, and defects are addressed by building tests to overcome them. System tests are run frequently to maintain a high standard of code quality (Yasvi et al., 2019).

Frameworks and Databases

The Python framework Django and the SQLite database were chosen for this project due to their alignment with the need for rapid development and ease of use. Django, a full-stack framework, offers built-in components for both frontend and backend development, enabling rapid iteration and scalability, which is crucial for larger projects (Duggal, 2022 & Diaz, 2024). Similarly, SQLite is a lightweight, serverless relational database ideal for development and testing phases, offering simplicity and ease of setup without the need for a server, making it perfect for quick iteration progress (Andor et al., 2022 & Simplilearn, 2023).

Previous Works

Recent research underscores the effectiveness of the Random Forest (RF) algorithm in detecting password attacks and its applicability to this project. Charmanas et al. (2023) demonstrated RF's ability to classify vulnerabilities with around 87% accuracy, proving its reliability for cybersecurity tasks. Additionally, Akbar et al. (2023) showed RF's capability to detect brute-force attacks in the MQTT protocol with significant accuracy, handling large datasets and extracting meaningful patterns. While the primary goal of this project is to evaluate RF's effectiveness rather than achieving high accuracy, these studies provide a strong foundation for using RF to recommend penetration testing tools based on user input, ensuring the project's success (Charmanas et al., 2023; Akbar et al., 2023).

METHODOLOGY

XP methodology is an agile software development approach that prioritizes adaptability, reactivity, and customer satisfaction. It follows five rules: planning, managing, designing, coding, and testing, fostering quality and reliability.

Workflow

The AI Recommendation Penetration Testing Tools for Password Attack project begins with users registering or logging in to access the dashboard, where they input their penetration testing requirements. The system, using the Random Forest algorithm, processes these inputs to recommend appropriate tools such as Nmap, Hydra, Medusa, and Wfuzz. Users can agree or disagree with the recommendations, providing feedback to improve the AI's accuracy. Results and feedback are saved to the database for future reference, allowing users to manage their penetration testing activities efficiently. The AI system continuously learns from user interactions to enhance future recommendations. Administrators log in to the system to manage user accounts and oversee user requirements and suggestions. They can edit or delete user information and feedback to maintain system quality and accuracy. Admins also monitor user satisfaction and adjust improve the AI's recommendations. This administrative oversight ensures the system remains up-to-date and effective, ultimately enhancing the overall user experience and security practices.

System Architecture

This project's system architecture aims for optimal performance and user satisfaction by incorporating multiple layers with specific tasks, ensuring seamless operation and effective interaction.

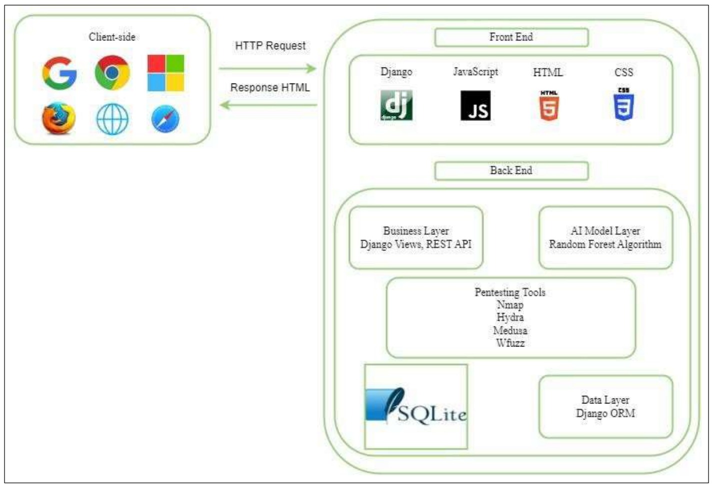


Figure 1.2 System Architecture of The Website

The system architecture, as shown in Figure 3.4, consists of multiple layers: the client-side (user interface), back-end (business logic), AI model layer, and data layer. The client-side, built with Django, JavaScript, HTML, and CSS, facilitates user interactions with an intuitive interface. The back-end, using Ubuntu Server 24.04 LTS, handles core operations and processes user requests, ensuring smooth coordination between the front end and data layer. The AI model layer employs the Random Forest Algorithm to analyze user inputs and recommend the best penetration testing tools for password attacks. The data layer, managed by Django ORM, stores essential information such as user details and tool suggestions in a lightweight database. When a user interacts with the system, the browser sends an HTTP request to the server, which processes the request and retrieves the necessary data from the database. The server then returns an HTML response to the user's browser. This architecture ensures users receive optimal recommendations for security testing tools efficiently and effectively, with each layer performing distinct functions to support the system's objectives.

RESULT AND DISCUSSION

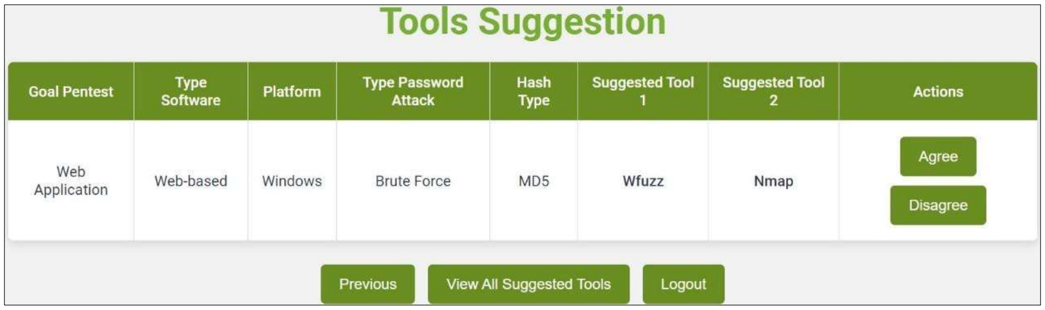
The project's primary aim was to evaluate the feasibility of using Random Forest for recommending penetration testing tools for password attacks. This chapter presents the findings, analysis, and testing procedures to ensure the reliability of the recommendation results, highlighting technical aspects and their implications for enhancing cybersecurity strategies.

Model Implementation

The implementation of the Random Forest model was crucial for developing the AI recommendation system for penetration testing tools. The 'scikit-learn' library's 'RandomForestClassifier' was used to create and train the model. The model utilized attributes like the goal of the pentest, type of software, platform, type of password attack, and hash type as input features to generate recommendations. Parameters such as 'n_estimators' and 'max_depth' were optimized for performance. The 'fit' method trained the model, while the 'predict' method provided tool recommendations based on new data. The 'scikit-learn' library facilitated rapid development and testing, aligning with the project's requirements for efficient and reliable tool recommendations.

Recommendation Interface

This part analysis of the website interface execution focused on evaluating how the application performed from both the client side and in the backend. At the client side, the user interaction within the interfaces like the result they gained, the response time and others, meanwhile on the backend the system performance has been reviewed. This analysis are important to check the inefficiencies of the system and improve it. Figure 1.3 below, shown the page of tools recommendation.



Tools Suggestion

Goal Pentest	Type Software	Platform	Type Password Attack	Hash Type	Suggested Tool 1	Suggested Tool 2	Actions
Web Application	Web-based	Windows	Brute Force	MD5	Wfuzz	Nmap	Agree Disagree

Figure 1.3 Tools Recommended

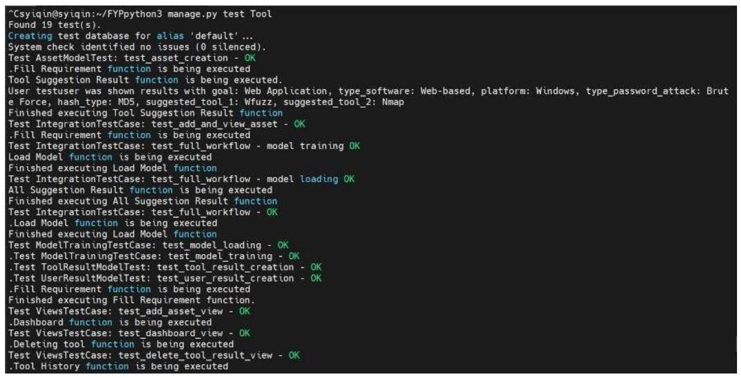
In this Figure 1.3 the recommendation pentest tool for password attack has been shown. That interface is for the client side. The user can give feedback towards those suggested result they got whether they agree or disagree. After the user give feedback then action column gone to make sure that user gave only once feedback. Then, user can choose to use the tool or view the history or logout.

Unit Testing

Unit testing involved testing individual components like the Asset model, Tool Result, and User Result model, all defined in the 'models.py' file. For example, the AssetModelTest class created initial data and verified correct asset creation, ensuring user requirements were saved correctly. The UserResultModelTest and ViewsTestCase classes similarly verified model and view functionality, confirming correct data handling and response to HTTP requests. The ModelTrainingTestCase class tested model training for the Random Forest algorithm, confirming successful model preparation.

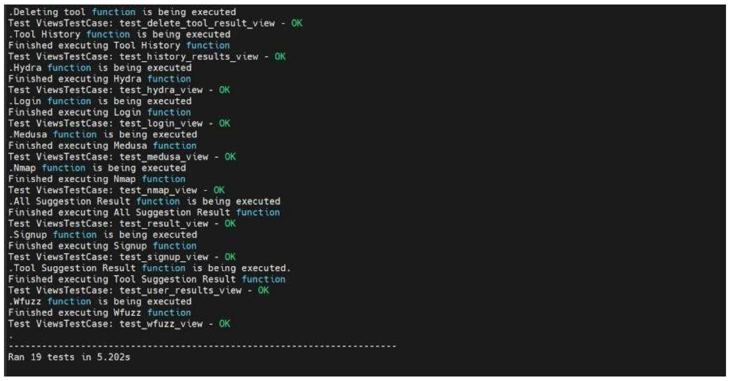
Integration Testing

Integration testing combined components to ensure proper interaction and data flow. The IntegrationTestingCase class verified system interactions like user login, authentication, and asset functionality. The testing process included adding assets, creating test data, training the model, and verifying successful integration and status codes. This comprehensive approach ensured the system's reliability and seamless operation, confirming all parts worked together without errors.



```
^Csyiqin@syiqin:~/FYPpython3 manage.py test Tool
Found 19 test(s).
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
Test AssetModelTest: test_asset_creation - OK
.Fill Requirement function is being executed
Tool Suggestion Result function is being executed.
User testuser was shown results with goal: Web Application, type_software: Web-based, platform: Windows, type_password_attack: Brut
e Force, hash_type: MD5, suggested_tool_1: Wfuzz, suggested_tool_2: Nmap
Finished executing Tool Suggestion Result function
Test IntegrationTestCase: test_add_and_view_asset - OK
.Fill Requirement function is being executed
Test IntegrationTestCase: test_full_workflow - model training OK
Load Model function is being executed
Finished executing Load Model function
Test IntegrationTestCase: test_full_workflow - model loading OK
All Suggestion Result function is being executed
Finished executing All Suggestion Result function
Test IntegrationTestCase: test_full_workflow - OK
.Load Model function is being executed
Finished executing Load Model function
Test ModelTrainingTestCase: test_model_loading - OK
.Test ModelTrainingTestCase: test_model_training - OK
.Test ToolResultModelTest: test_tool_result_creation - OK
.Test UserResultModelTest: test_user_result_creation - OK
.Fill Requirement function is being executed
Finished executing Fill Requirement function.
Test ViewsTestCase: test_add_asset_view - OK
.Dashboard function is being executed
Test ViewsTestCase: test_dashboard_view - OK
.Deleting tool function is being executed
Test ViewsTestCase: test_delete_tool_result_view - OK
.Tool History function is being executed
```

Figure 1.4 *Testing Result for Unit Testing*



```
.Deleting tool function is being executed
Test ViewsTestCase: test_delete_tool_result_view - OK
.Tool History function is being executed
Finished executing Tool History function
Test ViewsTestCase: test_history_results_view - OK
.Hydra function is being executed
Finished executing Hydra function
Test ViewsTestCase: test_hydra_view - OK
.Login function is being executed
Finished executing Login function
Test ViewsTestCase: test_login_view - OK
.Medusa function is being executed
Finished executing Medusa function
Test ViewsTestCase: test_medusa_view - OK
.Nmap function is being executed
Finished executing Nmap function
Test ViewsTestCase: test_nmap_view - OK
.All Suggestion Result function is being executed
Finished executing All Suggestion Result function
Test ViewsTestCase: test_result_view - OK
.Signup function is being executed
Finished executing Signup function
Test ViewsTestCase: test_signup_view - OK
.Tool Suggestion Result function is being executed.
Finished executing Tool Suggestion Result function
Test ViewsTestCase: test_user_results_view - OK
.Wfuzz function is being executed
Finished executing Wfuzz function
Test ViewsTestCase: test_wfuzz_view - OK
.
----------------------------------------------------------------------
Ran 19 tests in 5.202s
```

Figure 1.5 *Result for Integration Testing*

Based on Figure 1.4 and Figure 1.5, the unit testing and integration system for this project was successfully done. The system was working smoothly because all the function correctly functioned.

CONCLUSION

Extreme Programming (XP) is an agile software development methodology that aims to enhance flexibility and responsiveness in the development process. This project successfully developed an AI-based recommendation system for selecting penetration testing tools for password attacks using the Random Forest algorithm. The project significantly reduced the time and expertise required for tool selection, demonstrating the potential of AI in cybersecurity. XP encourages continuous improvement and adaptability, focusing on simplicity, continuous integration, and simplicity. The five fundamental rules of XP methodology include planning, managing, designing, coding, and testing. Planning involves identifying, ranking, and estimating user stories, while managing involves iterative and incremental development with planned small releases and continuous progress monitoring. Designing prioritizes simplicity, making incremental decisions to enhance maintainability and minimize flaws. Testing involves continuous integration and automated unit, integration, and acceptance tests to ensure reliability and prevent errors.

REFERENCES

Akbar, G. M., Hariyadi, M. A., & Hanani, A. (2023). Detection Of Bruteforce Attacks On The MQTT Protocol Using Random Forest Algorithm. *Internet of Things and Artificial Intelligence Journal*, *3*(3), 250-272. https://doi.org/10.31763/iota.v3i3.630

Charmanas, K., Mittas, N., & Angelis, L. (2023). Exploitation Of Vulnerabilities: A Topic-Based Machine Learning Framework For Explaining And Predicting Exploitation. *Information*, *14*(7), 403. https://doi.org/10.3390/info14070403

Diaz, D. (2024, April 2). *25 Python Frameworks Worth Learning*. Kinsta®. https://kinsta.com/blog/python-frameworks/

Duggal, N. (2022, February 18). *What Are Frameworks in Python? Know Top 5 Python Frameworks*. Simplilearn.com. https://www.simplilearn.com/python-frameworks-article

Lifandali, O., Abghour, N., & Chiba, Z. (2023). Feature Selection Using A Combination Of Ant Colony Optimization And Random Forest Algorithms Applied To Isolation Forest Based Intrusion Detection System. *Procedia Computer Science*, 220, 796-805. https://doi.org/10.1016/j.procs.2023.03.106

Meier, K. (2023, July 20). The Simple Guide To Startup Product Development: From Ideation To Launch. *The Startup Chat with Steli & Hiten*. https://thestartupchat.com/startup-product-development/

Saber, V., ElSayad, D., Bahaa-Eldin, A. M., & Fayed, Z. (2023). Automated Penetration Testing, A Systematic Review. *2023 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*. https://doi.org/10.1109/miucc58832.2023.10278377

Schonlau, M., & Zou, R. Y. (2020). The Random Forest Algorithm For Statistical Learning. *The Stata Journal: Promoting communications on statistics and Stata*, *20*(1), 3-29. https://doi.org/10.1177/1536867x20909688

Shrivastava, A., Jaggi, I., Katoch, N., Gupta, D., & Gupta, S. (2021). *A Systematic Review On Extreme Programming* [Paper Presentation]. Journal Of Physics: Conference Series. https://iopscience.iop.org/article/10.1088/1742-6596/1969/1/012046/pdf

Simplilearn. (2023, January 12). *What Are The Various Types of Databases?* Simplilearn.com. https://www.simplilearn.com/tutorials/dbms-tutorial/what-are-various-types-of-databases

Yasvi, M. A., Yadav, K. S., & Shubhika. (2019). Review On Extreme Programming-XP. https://www.researchgate.net/publication/332465869_Review_On_Extreme_Programming-XP

Zhang, C., Hu, C., Xie, S., & Cao, S. (2021). Research On The Application Of Decision Tree And Random Forest Algorithm In The Main Transformer Fault Evaluation. *Journal of Physics: Conference* Series. https://iopscience.iop.org/article/10.1088/1742-6596/1732/1/012086/pdf