Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

Hybrid Random Forest And Isolation Forest Model For Zero Day Vulnerability Detection

Anjitha S
Department of Computer Science
College of Applied Science, Anchunadu, Kanthalloor
Kerala, India

Anandhakrishnan R
Department of Computer Science
College of Applied Science, Anchunadu, Kanthalloor
Kerala, India

ABSTRACT **- In the modern digital era, software applications are widely used in sectors such as finance, healthcare, communication, and government services. As software systems become more complex, the risk of security vulnerabilities increases. Attackers constantly search for weaknesses in software systems to exploit them for unauthorized access or data theft. One of the most serious threats in cybersecurity is the presence of Zero Day vulnerabilities. A Zero Day vulnerability is a security flaw in software that is unknown to developers or security experts when it is discovered and exploited by attackers. Since these vulnerabilities are not yet documented and no patches are available, traditional security tools that rely on known vulnerability signatures often fail to detect them. Existing vulnerability detection methods typically involve manual code review or rule-based static analysis tools. This paper presents a Hybrid Random Forest and Isolation Forest Model for Zero Day Vulnerability Detection. The proposed system not only improves the efficiency of vulnerability detection but also assists developers in writing more secure software by providing detailed insights into potential security weaknesses in their code.**

Keywords - Cybersecurity, Zero Day Attack, Vulnerability Detection, Machine Learning, Isolation Forest, Random Forest, Static Code Analysis.

I. INTRODUCTION

As digital transformation accelerates, the size, complexity, and interconnectivity of software systems continue to increase. it also significantly expands the attack surface for cyber threats. Software vulnerabilities are weaknesses or flaws in a system that can be exploited by attackers to disrupt services or compromise system integrity Security weaknesses happen when software is built with mistakes. Such as flaws in the original plan or code. While these errors might seem small, a hacker can use them to break into a system and cause serious damage. Among various types of vulnerabilities, Zero Day vulnerabilities represent one of the most dangerous categories. A Zero Day vulnerability refers to a previously unknown security flaw in software that is discovered and exploited by attackers before developers or security professionals are aware of it. Traditional vulnerability detection systems rely primarily on signature-based detection and rule-based analysis. Signature-based systems compare code or system behaviour against a database of known vulnerability patterns. While effective in detecting previously identified threats, these systems fail to recognize new, unknown, or modified attack techniques. Additionally, maintaining up-to-date signature databases requires continuous monitoring and updates, making these systems reactive rather than proactive. Static code analysis tools analyze source code without executing it, searching for predefined insecure coding patterns. Although useful, these tools often generate a large number of false positives and lack the intelligence to adapt to new vulnerability patterns. Machine learning models can analyze large volumes of data, learn underlying patterns, and identify deviations from normal behaviour. Instead of relying solely on predefined signatures, AI-based systems learn from historical code data and detect anomalies that may indicate potential vulnerabilities. By identifying unusual patterns and classifying risk levels, such systems enable proactive and adaptive cybersecurity defense mechanisms. Therefore, an integrating AI and Machine Learning ML techniques into vulnerability detection systems represents a significant advancement in modern cybersecurity research and practical application.

II.LITERATURE REVIEW

Several researchers have explored machine learning methods for software vulnerability detection and cybersecurity professionals have proposed multiple approaches for identifying software vulnerabilities and malicious activities. As software systems become more complex, the risk of security vulnerabilities increases. Among these threats, Zero Day vulnerabilities are particularly dangerous because they exploit previously unknown flaws in software before developers or security teams become aware of them. Since no predefined signatures or patches exist for such vulnerabilities, traditional detection systems often fail to identify them. Most conventional vulnerability detection tools rely on signature-based or rule-based methods, where software code is analyzed and compared with known vulnerability patterns stored in datasets. Although these methods are effective for identifying previously known threats, they cannot detect new or unknown vulnerabilities. This limitation creates a significant security gap in modern software development environments.

When, during development, developers must review and test the source code to ensure that security vulnerabilities do not exist within the application. However, manual code reviews and traditional vulnerability detection tools may not always detect newly emerging threats, especially zero day vulnerabilities that have not yet been documented in vulnerability databases. Despite significant improvements, AI based vulnerability detection software systems still face limitations including adversarial attacks and difficulties in detecting new attacks and including high false-positive rates and reduced effectiveness in identifying unknown

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

vulnerabilities. The proposed system extends existing research by introducing a cost-effective and efficient machine learning based vulnerability detection system. Isolation Forest gained attention because of its ability to detect unusual patterns efficiently in large datasets. Similarly, Random Forest became widely adopted for classification problems due to its high prediction accuracy and robustness.

III.PROBLEM STATEMENT

Most traditional tools rely heavily on known vulnerability signatures. They compare input code against predefined vulnerability databases. This approach makes them ineffective against new or previously unseen attack patterns. These systems require continuous rule and signature updates. As new vulnerabilities are discovered, developers must manually update detection rules. This reactive approach delays protection and leaves systems exposed during the update gap. Signature-based systems cannot identify variations that do not exactly match stored patterns. Many existing static analysis tools produce a high rate of false positives. traditional systems lack adaptive learning capability. They do not improve automatically based on new data or evolving attack techniques. Without learning mechanisms, their effectiveness remains limited.

Despite the availability of numerous vulnerability detection tools, several critical limitations remain unresolved in existing systems. Therefore, there is a strong need for an adaptive vulnerability detection framework capable of identifying suspicious coding behaviour automatically using machine learning techniques.

IV.OBJECTIVES OF THE PROPOSED SYSTEM

The project aims to demonstrate the practical application of machine learning techniques in strengthening cybersecurity systems.

- To design an AI-driven system for detecting zero day vulnerabilities
- To implement zero day vulnerabilities detection using the Isolation Forest algorithm, enabling the system to identify unusual coding patterns that deviate from normal behaviour.
- To implement vulnerability severity classification using the Random Forest algorithm, categorizing detected issues into levels such as Low, Medium, or High risk.Units
- To generate a confidence score that indicates the reliability and probability of the prediction.
- To develop a secure web-based interface that allows user registration, login, and code submission.
- To implement preprocessing and feature extraction mechanisms that convert source code into meaningful numerical representations suitable for machine learning models. state the units for each quantity that you use in an equation.
- To generate structured and understandable vulnerability reports that assist developers in improving code security.
- To minimize false positives and improve detection accuracy compared to traditional rule-based tools.

V.PROPOSED METHODOLOGY

The proposed system performs source code preprocessing, feature extraction, anomaly detection using Isolation Forest, and vulnerability classification using Random Forest. The framework identifies suspicious coding behaviour and generates structured vulnerability reports.

A. Data Collection

The dataset used in this research was gathered from open-source repositories, publicly available vulnerability datasets, and cybersecurity research platforms. The collected data included examples of secure code and vulnerable source code structures

B. Source Code Preprocessing

Before analysis, the submitted source code undergoes preprocessing to remove unnecessary elements such as comments, whitespace, redundant symbols, and formatting inconsistencies. Tokenization and normalization are then performed to prepare the data for feature extraction.

C. Feature Extraction

Feature extraction converts source code into numerical representations suitable for machine learning algorithms. Features include token frequency, function usage patterns, input validation structures, suspicious API calls, and insecure coding patterns.

D. Anomaly Detection Using Isolation Forest

Isolation Forest is used to identify abnormal coding behaviour within source code. The algorithm isolates unusual patterns that differ significantly from normal programming structures. Such anomalies may indicate zero day vulnerabilities or suspicious code behaviour. Some Common Mistakes.

E. *Vulnerability Classification Using Random Forest*

Random Forest is used to classify detected vulnerabilities into different severity levels such as Low, Medium, and High. The classifier improves prediction reliability and reduces false-positive rates.

F. Report Generation.

The system generates a structured vulnerability report containing:

- Vulnerability type
- Severity level
- Confidence score
- Vulnerable line number

G. Suggested security information

The generated reports assist developers in identifying and resolving vulnerabilities efficiently.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

VI. SYSTEM ARCHITECTURE

The architecture of the proposed framework consists of User Interface Module, Source Code Upload Module, Preprocessing Engine, Feature Extraction Module, Machine Learning Analysis Module, Vulnerability Classification Module, Report Generation Module.

The frontend interface was developed using HTML, CSS, and JavaScript. Backend services were implemented using FastAPI and Python, while machine learning functionality was developed using Scikitlearn, Pandas, and NumPy.

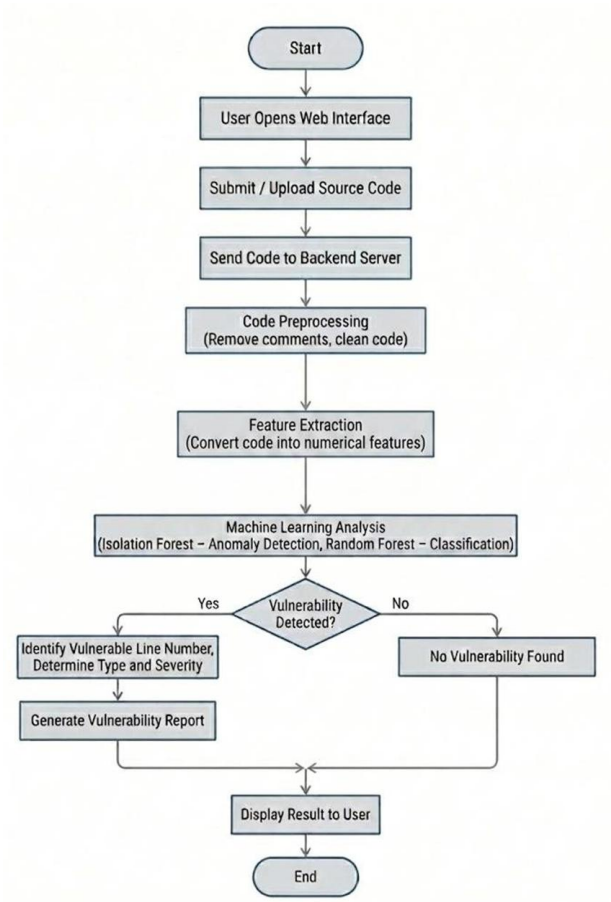

Fig 1. Working Flow of Hybrid Random Forest and Isolation Forest Model for Zero Day Vulnerability Detection

VII. IMPLEMENTATION DETAILS

Python was selected as the primary programming language because of its flexibility and extensive machine learning library support. Fast API was used to implement backend communication services and API endpoints. The implementation process involved: Dataset preprocessing, Feature engineering, Model training and testing, Backend API integration, Vulnerability analysis implementation, Result visualization, Automated report generation.

The implementation of the system was carried out using modern programming languages and tools suitable for machine learning and web application development. Selecting an appropriate implementation environment is essential for ensuring smooth development and system performance. The backend of the system was developed using Python, which provides extensive support for machine learning, data processing, and web application development. Python was selected because of its simple syntax, readability, flexibility, and wide availability of libraries that support artificial intelligence and data analysis. Python also supports rapid development, making it suitable for building intelligent systems. The Fast API framework was used to build the backend API that manages communication between the frontend interface and the machine learning module. Fast API allows the system to process requests efficiently and supports asynchronous operations, which improves the speed of handling multiple user requests. It also provides automatic validation of input data and built-in API documentation features. The backend of the system was developed using Python, which provides extensive support for machine learning, data processing, and web application development. Python was selected because of its simple syntax, readability, flexibility, and wide availability of libraries that support artificial intelligence and data analysis. Python also supports rapid development, making it suitable for building intelligent systems. The FastAPI framework was used to build the backend API that manages communication between the frontend interface and the machine learning module. Fast API allows the system to process requests efficiently and supports asynchronous operations, which improves the speed of handling multiple user requests. It also provides automatic validation of input data and built-in API documentation features.

Machine learning models used for vulnerability detection were implemented using the Scikit-learn library, which provides reliable algorithms for classification and anomaly detection tasks. The algorithms selected for this system include:

- Isolation Forest for anomaly detection
- Random Forest for classification of vulnerability severity

These algorithms were trained using prepared datasets containing examples of secure and vulnerable code patterns. The trained models were tested using validation datasets to ensure their accuracy before integration into the system. Data preprocessing and feature extraction tasks were carried out using Pandas and NumPy, which provide efficient tools for handling structured data and performing numerical computations. These libraries allowed the system to process large amounts of code data efficiently. The frontend interface of the system was developed using HTML, CSS, and JavaScript, allowing users to interact with the system through a web browser. The web-based design ensures that users can access the system without installing additional software.

VIII. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed framework was tested using multiple source code samples containing both secure and vulnerable programming structures.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

OUTPUTS

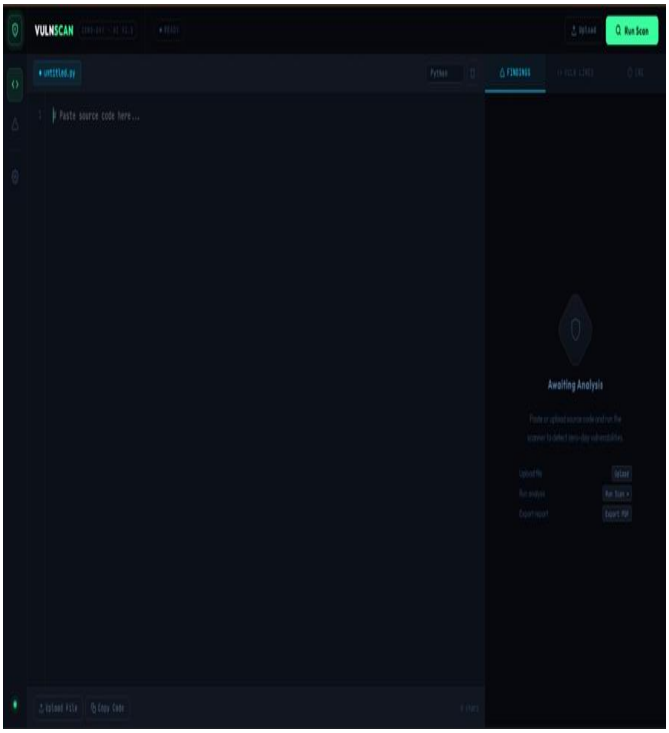

Fig 2 . Zero-Day Vulnerability Dashboard

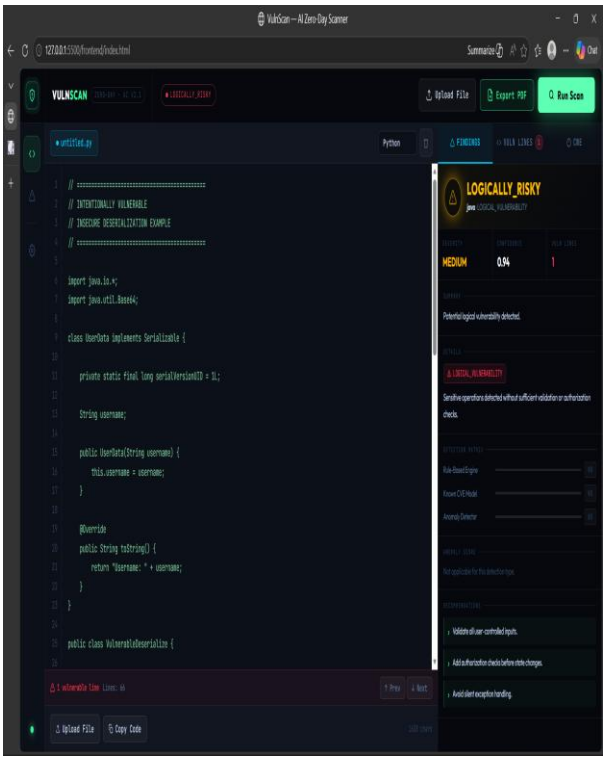

Fig 4. Vulnerability Inspection Panel(Medium)

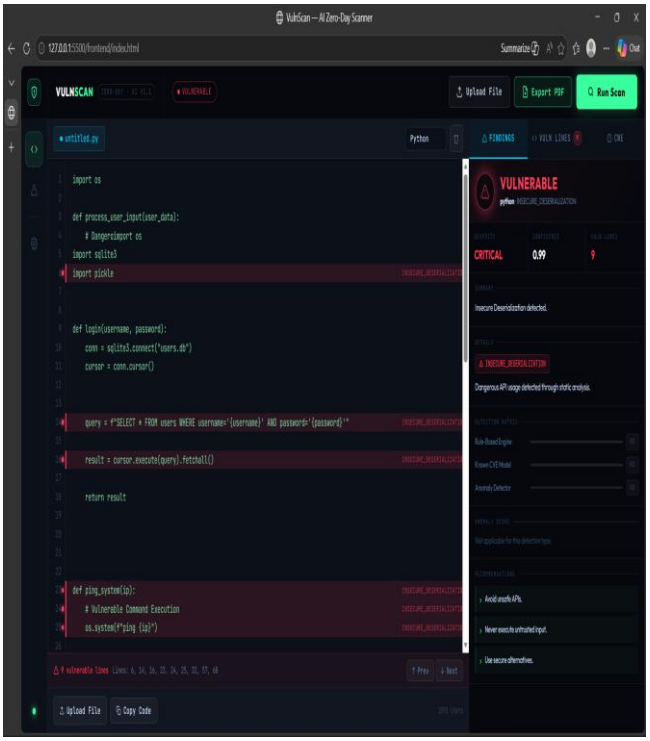

Fig 3. Vulnerability Inspection Panel(Critical)

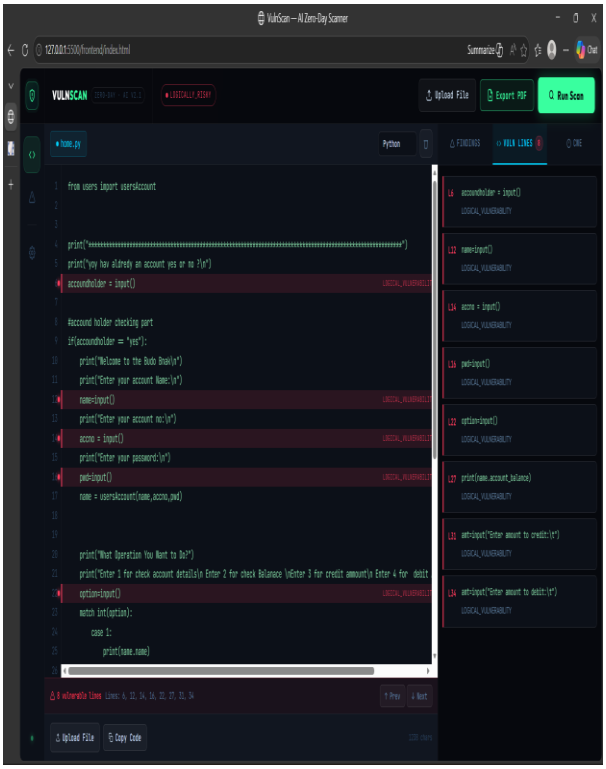

Fig 5. Vulnerability Line Analysis

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

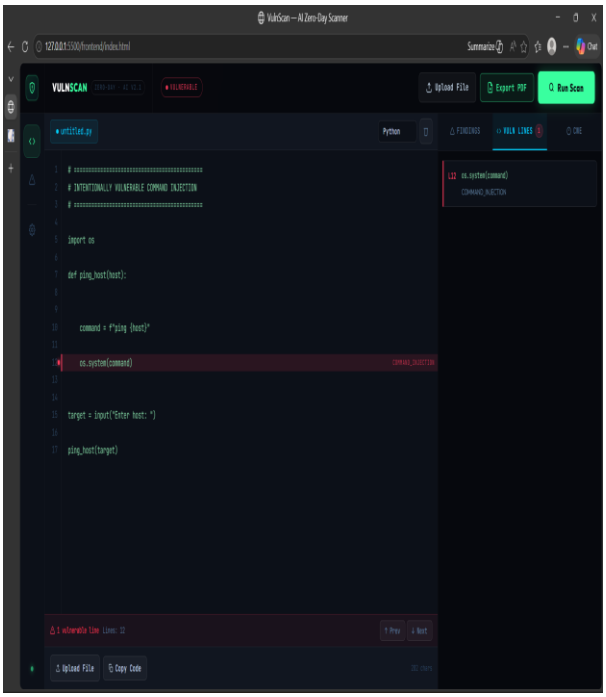

Fig 6. Vulnerability Line Analysis (Command Injection)

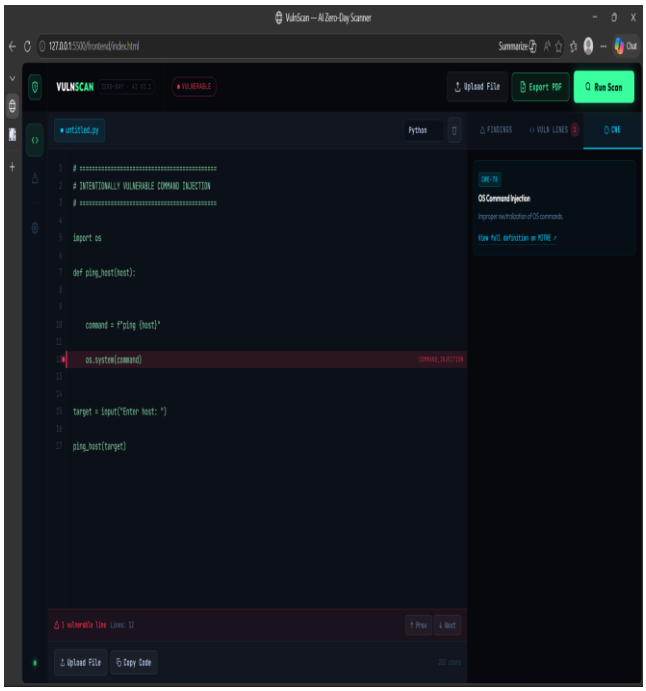

Fig 8. CWE Vulnerability Reference (OS Command Injection)

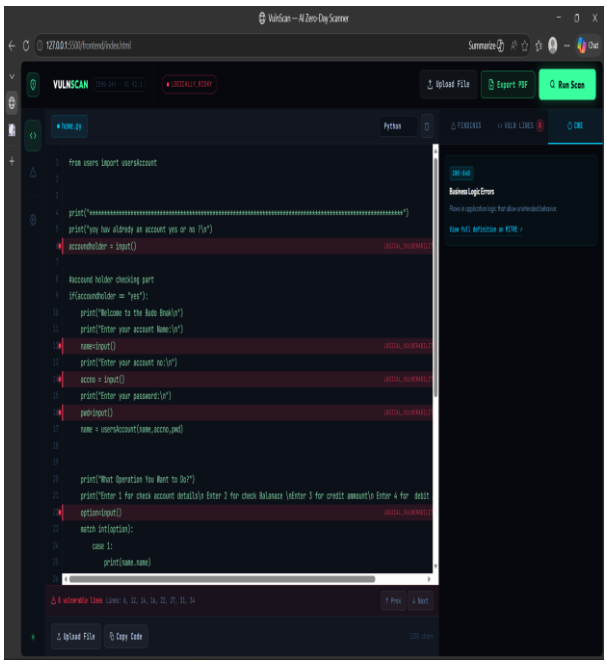

Fig 7. CWE Vulnerability Reference (Business logic errors)

IX. ADVANTAGES OF THE PROPOSED SYSTEM

The proposed system provides multiple advantages compared with traditional vulnerability detection approaches. The proposed approach offers several advantages:

- Improved vulnerability detection accuracy
- Reduced false positives
- Automated analysis process
- Faster detection capability
- Scalable architecture
- Better software security support

The results demonstrate that machine learning techniques can significantly enhance traditional static analysis systems.

X. FUTURE ENHANCEMENTS

The AI-Based Zero Day Vulnerability Detector developed in this project provides an effective solution for detecting potential vulnerabilities in source code using machine learning techniques.

One of the major areas for future enhancement is the support for multiple programming languages. The current version of the system may focus on analyzing specific programming languages such as python and java or limited code patterns. In the future, the system can be extended to support additional programming languages.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 05 , May - 2026

The system can also be enhanced by implementing dynamic code analysis in addition to static analysis. While static analysis examines source code without executing it, dynamic analysis monitors program behaviour during execution. Combining both static and dynamic analysis techniques will provide a more comprehensive vulnerability detection approach and improve detection reliability. Another possible enhancement is the integration of the system with Continuous Integration and Continuous Deployment CI/CD pipelines.

Future improvements may also include the development of an automated vulnerability fixing system. In addition to detecting vulnerabilities, the system can suggest recommended solutions or code modifications to fix detected security issues. Cloud-based deployment is another possible enhancement that can improve system scalability and accessibility. By deploying the system on cloud platforms, large-scale code analysis can be performed efficiently, and users from different locations can access the system easily.

Finally, the system can be enhanced by improving the dataset used for training machine learning models. Increasing the size and diversity of training datasets will improve detection accuracy and enable the system to identify a wider range of vulnerabilities. Continuous model training using updated datasets will ensure that the system remains effective against emerging security threats.in conclusion, These enhancements can further improve the accuracy, scalability, and real-world applicability of the system.

XI. CONCLUSION

This paper presented a Hybrid Random Forest and Isolation Forest Model for Zero Day Vulnerability Detector for intelligent source code analysis. The proposed framework integrates Isolation Forest and Random Forest algorithms to improve vulnerability detection capability while reducing false positives. The system provides a scalable and efficient cybersecurity solution for modern software applications.

XII. REFERENCES

[1] I. Sommerville, Software Engineering, 10th ed. Pearson Education, 2016.
[2] R. S. Pressman, Software Engineering: A Practitioner's Approach, 8th ed. McGraw-Hill Education, 2015.
[3] T. M. Mitchell, Machine Learning. McGraw-Hill Education, 1997.
[4] S. Raschka and V. Mirjalili, Python Machine Learning. Packt Publishing, 2019.
[5] OWASP Foundation, "OWASP Top 10:2021-The Ten most critical Web Application Security Risks," 2021.
[6] Python Software Foundation," Python Documentation," [online]Available at: *https://docs.python.org*
[7] Scikit-learn Developers," Scikit-learn Documentation," [online]Available at: *https://scikit-learn.org*
[8] F. Chollet, Deep Learning with Python. Manning Publications, 2018.
[9] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[10] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proceedings of the IEEE International Conference on Data Mining, 2008, pp. 413–422.
[11] A. Géron, Hands-On Machine Learning with Scikit- Learn, Keras, and TensorFlow. O'Reilly Media, 2019.