International Journal of Artificial Intelligence, Data Science, and Machine Learning
Grace Horizon Publication | Volume 6, Issue 2, 73-82, 2025
ISSN: 3050-9262 | https://doi.org/10.63282/3050-9262.IJAIDSML-V6I2P108

Original Article

Securing Modern Web Applications Using AI-Driven Static and Dynamic Analysis Techniques

Sandeep Phanireddy
Sr, Product Security Engineer, USA.

Received On: 05/03/2025 ***Revised On: 15/03/2025*** ***Accepted On: 31/03/2025*** ***Published On: 20/04/2025***

Abstract: *The application layer in the modern web, developed for various purposes such as banking, social networking, etc., is widely exposed to cyber threats due to loopholes in application architecture. To address these issues, it has been found that the incorporation of AI in security analysis has been quite effective. In this paper, the author scrutinizes the manner in which static and dynamic analysis methods propelled by artificial intelligence can strengthen the security of the current web-intensified applications. Here, we see the very essence of modern web applications and identify the significant increase in the number and depth of threats. We then provide a comparative analysis of traditional and AI methods for detecting vulnerabilities. What pertains to static analysis that analyzes code without executing it is discussed regarding applying machine learning classifiers and code understanding based on NLP. On the other hand, dynamic analysis that involves determining the behavior of an application in operation can rely on reinforcement learning and anomaly detection. In this paper, we propose a framework that incorporates both approaches which is well demonstrated through an actual e-commerce environment. Given outcomes suggest a gain in the number of birds detected, minimized false alarms, and quicker response time. It also covers implementation issues such as the lack of datasets and generalizing and incorporating the model into the DevSecOps pipeline. In conclusion, incorporating AI-based analysis provides an active and elastic approach to safeguard web applications against existing and arising hazards.*

Keywords: *Web Security, Artificial Intelligence, Static Analysis, Dynamic Analysis, Machine Learning, Vulnerability Detection.*

1. Introduction

1.1 The Need for AI in Security Analysis

With the continuous development of complex web applications and their increased importance to businesses, such kind of methods of security analysis are becoming even more important. [1-4] Procedures involved in the past, including manual code reviews or the application of rules-based vulnerability scanners, are inadequate to meet today's growing threats. This section focuses on identifying factors that justify the use of AI in security analysis to ascertain why it is crucial in the field.

Figure 1: The Need for AI in Security Analysis

- **Growing Complexity of Web Applications:** Web applications are characterized by their increasing complexity, interactivity and functionality. This is due to other features like complicated coding structures, layered services, and high third-party incorporation mostly difficult to diagnose by hand. As new technologies like microservices, APIs, and serverless are introduced, the attack vector has shifted. Moreover, traditional security tools do not evolve with such complex structures of the systems and are incapable of evaluating their security efficiently. Thus, AI can handle these complexities on the go and is able to predict new and other forms of risks as they develop over time.
- **Speed and Volume of Development Cycles:** When it comes to development in the contemporary world, people follow the idea of 'time is money.' When it comes to software development, segregated phases of development are a thing of the past, with the current paradigm embracing Continuous Integration/Continuous Deployment (CI/CD) pipes. There is barely any space left for manual security testing, which slows release cycles and may lead to more susceptibilities being missed. Analysts found advantages in integrating AI-driven security analysis in these security pipelines. This automation enables security to maintain parity with development to identify and report issues without slowing down development.
- **The Evolving Threat Landscape:** It is also suggested that threats in cyberspace are advancing and tend to employ methods like machine learning, social engineering, and zero-day attacks. These emerging threats sometimes call for detection mechanisms different from the norm, which involves simple rules-based detection that can only be effective when used to identify known threats. AI is particularly good at discovering new patterns of attack and potential weaknesses that may be latent and not easily discernible using other techniques. AI can be used to identify new trends in hacking, which makes this technology quite effective in the fight against ever-evolving hackers.
- **Scarcity of Skilled Security Professionals:** This is a problem because there is a shrinking pool of qualified personnel in the cybersecurity industry, and this means that organizations cannot have adequate personnel for deeper analysis of security issues. As for manual testing, it demands considerable effort and experienced specialists, which are relatively few. This can be useful to close the gap and facilitate the security analysis process by automating many facets without additional human resources. Automating a large part of the work may be helpful to security specialists who can be relieved from routine tasks and devote their time to important cases that call for their attention.
- **The Need for Proactive Security Measures:** Traditionally, security for applications on the web has been of a reactive type, where measures are taken only after incidents happen or are discovered. This makes them take a reactive approach that exposes applications to cyberattacks for a long time. In contrast, AI-based security analysis is more preventative analysing the system to find out what can go wrong in a system. Thus, AI can foresee possible threats and prevent them by having algorithms to sort numbers, characterizing patterns and behaviors, and using machine learning. This also ensures that businesses adopt preventive measures against security breaches in the development phase, thus mitigating the vulnerability of such instances and the cost incurred in handling them. Altogether, such factors as the increasing intricacy of Web applications, pressure on development time, development of threats, shortage of skilled workers, and the necessity to have active safety measures mean that AI is becoming critical in security analysis. AI results in enhanced capabilities to meet contemporary security issues, allowing organizations' web applications to be safe against malicious actors.

1.2 AI-Driven Static and Dynamic Analysis Techniques

Static and dynamic analysis methodologies are strong security solutions in modern web applications through machine learning and artificial intelligence. [5,6] Static analysis, on the other hand, refers to inspecting a program without actually running it to find syntax bugs, insecure coding practices or weakened security breaches such as SQL injection. AI also improves static analysis by utilising algorithms from the sphere of machine learning to search for patterns of behavior in a code that can be considered a vulnerability. It can also automatically rank vulnerabilities depending on their criticality and does not necessarily require one to assess. Static analysis's strength in AI is its capacity to ingest large amounts of code quickly and then flag the probability of a mistake that might have otherwise escaped code review. Dynamic analysis, in contrast with static analysis, is performed at the runtime to check for the presence of issues that only become apparent during the actual use of the developed application, for example, when a user tries to enter the application with an appropriate password or if the access rights within the application are not properly implemented.

AI underlines dynamic analysis's effectiveness by allowing it to be conducted in real-time and adaptive mode. During runtime, the AI-based tools can monitor the behaviour of an application and then, based on this information, use predictive analysis to identify signs of a potential weak link in the security system. They can also model advanced attacks, which may not be displayed by the other means typical for security systems. Static and dynamic analysis work in union to present several advantages of AI; AI becomes the way of obtaining the synchronous view of the referred application's tendencies both at the code level and the runtime. These AI-based approaches help identify intricate problems and improve the positives while ensuring continuous security integration into the coding procedures to

rectify the issues. This is more than just a reactive approach, making it a robust, scalable and proactive measure that helps organizations prepare for new threats.

2. Literature Survey

2.1 Static Analysis Techniques

Static analysis entails analyzing source code without running it or profiling its behavior. A set of tools are used in the industry, like SonarQube, Fortify, and Checkmarx for identifying security vulnerabilities, code smells, and compliance. They all use the rule-based approach and work based on matching pre-defined templates by checking for mistakes throughout the development cycle. [7-11] Although they are not immune from a high false positive rate whereby, the reliability of the developers is easily compromised and therefore, important cues are missed. Furthermore, some competencies in static analysis tools, such as asynchronous JavaScript or dependency injection in Angular and Spring, cause issues in current development paradigms and frameworks, which restrains their efficiency in contemporary conditions.

2.2 Dynamic Analysis Techniques

DAST is the process of testing the application in a runtime environment and look for vulnerabilities that are not exposed if the Program testing is done at static state. OWASP ZAP and Burp Suite are successful tools for emulating real attack scenarios on web applications; some of the standard attacks that may be discovered include SQL injection attacks, cross-site scripting, and improper session management. The above tools are useful for runtime analysis but often do not know how the application works and what control flows suit it. Another limitation of using DAST tools is that they can easily get stuck in a routine where they run through the same patterns over and over until a problem is identified, even when better options exist because they do not have memory or intelligence for analysis over time.

2.3 AI in Static Analysis

New approaches to static analysis have brought semantic sensitivity and context into the picture with the help of AI into the picture. A distinctive example is DeepBugs here, the word embeddings and the neural networks are used to identify bugs with factors related to the semantics of tokens in source code. DeepBugs and similar models learn the code as natural language and are able to identify some peculiarities that rule-based approaches could not detect. By incorporating deep learning data, AI-based systems achieve higher precision and less false results, but that heavily depends on the quantity and quality of labeled datasets.

2.4 AI in Dynamic Analysis

Such an application of artificial intelligence in dynamic analysis is mainly through Anomaly detection methods and reinforcement learning. These include autoencoders, which can be trained on normal application behavior and then used to detect anomalies, including a security breach or software problem. Reinforcement learning agents have been evaluated to test the running system, teach the testing strategies to the learners, and learn the efficient strategies according to the specific application environment. A promising set of directions in AI patterns involves improving the responsiveness and the quality of detection of such bugs, although these are currently considered as research prototypes and scale-up and integration into the contemporary development processes pose some concerns.

2.5 Gaps Identified

Although work has been done combining static and dynamic analysis, integration stays relatively modest in numerous applications. This is perhaps a two-pronged approach that could be beneficial in getting better security insights in terms of both codes and the actual runtime behavior. Furthermore, most of the AI tools are in research and not widely used in the real world due to some concerns regarding reliability and interpretability and because they may require tuning for a specific area. Another major shortcoming is that there is no consistent set of datasets and performance measures to compare and compare tools and technologies. To fill in the given gaps more detailed and deeper understanding of this problem is important for further improvement of approaches to secure software development.

3. Methodology

3.1 Framework Overview

In this section, we propose a new generic approach that combines static and dynamic analysis concepts with the help of Artificial Intelligence (AI) algorithms. This approach is an attempt to improve on the limitations of the traditional approaches of static code inspection and runtime behavior analysis. To enhance the credibility and potential of the given framework, it can be underpinned by AI functionality, which can assist the process of a vulnerability scan, minimize the number of false positives, and make effective recommendations based on context. [12-16] The hybrid system combines three main modules, each with a different function, but the outcomes obtained are useful for performing security tests.

- **Source Code:** The source code is the first input to the hybrid framework. This is where the static analysis module starts and searches through the program for threats, errors, omissions, or security weaknesses. It is worth noting that most of the time, the source code is a static view of the application; it doesn't contain any runtime data, but it does contain a lot of patterns that can be marked problematic in accordance with certain defined rules, or learned by an AI model. This code can then be parsed by the framework to be input into the Static AI Module, where the statistical and machine learning algorithms are used to detect any risks as per past identifiable trends and any risks identified from contextual analysis of the code.

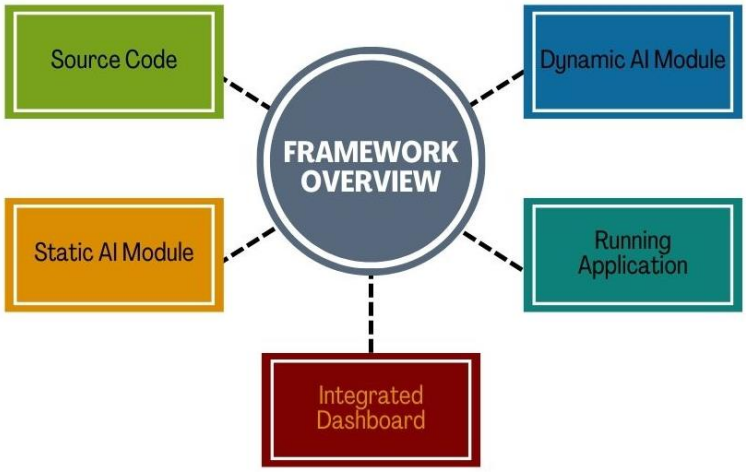

Figure 2: Framework Overview

- **Static AI Module:** Static AI module is the first core combined in the hybrid framework of the proposed system. Examining the application's source code without running it while incorporating concepts such as artificial intelligence, machine learning, and deep learning are other critical aspects of this type of analysis. While numerous traditional tools, such as PVS-Studio, work with strictly defined rules, the Static AI Module applies artificial intelligence to identify vulnerabilities that may have been in the past to anticipate potential problems in the source code. Thus, by understanding the meaning of the code snippets and recognizing the semantic patterns, especially the intricate relations between various sequences, the AI module can identify a potential flaw, a security issue or suboptimal code practices in a slice of code that might go unnoticed by humans. It can also evolve to accommodate the latest programming frameworks and paradigms, making it more efficient and versatile.
- **Integrated Dashboard:** The Integrated Dashboard, also known as the control panel, serves as the command center for monitoring and controlling the analysis process. It aggregates and presents the results from both the Static AI Module and the Dynamic AI Module to make them easily understandable to the developers, security teams, and stakeholders, all at one centralized place where they want to be. Overall, such a dashboard is management-oriented and dynamic and presents overviews of identified threats, the calculated level of risk, and suggested measures in order to facilitate the prioritization of the remediation based on risk's qualitative characteristics. However, there can be real-time analysis statistics, current threats and alerts observed while dynamic analysis is in progress and historical statistics to evaluate the application's safety at a definite time. They suggest that the integrated nature of this dashboard helps to provide synthesis and support decision-making across the Software Development Life Cycle.
- **Running Application:** The Running Application component describes the application's version in production or an environment in which the software can be tested as it would be used in actual use. This is possible through the running application that the Dynamic AI Module interfaces with the system in order to monitor the dynamic behavior. DAST is an example of dynamic analysis tools that evaluate the application's runtime environment to detect flaws that may not easily be seen from the code. These aspects include session management, input validation, and security misconfigurations that arise when the application is run. The real-time data is fed into the Dynamic AI Module incorporated in the fabric of the running application, where anomalies and correlations are identified and potential threats predicted based on context.
- **Dynamic AI Module:** The Dynamic AI Module parallels the Static AI Module, which analyses the application's behaviour during runtime. This module employs artificial intelligence and machine learning processes, including anomaly detection, reinforcement learning, and behavioural analysis, to detect security issues that are not detectable in a static analysis. For example, machine learning models can learn how an application is supposed to behave, what kind of traffic it is normally supposed to generate, or which APIs it should be calling and then look for the next unexpected activity or traffic that may be a sign of an attack or vulnerability. This kind of analysis is perfect for identifying threats in a real environment, while some other passive approaches might not discover threats during runtime. This is because the Dynamic AI Module is based upon previous incidents, real-time learning, and behavioural changes of application to proactively induce threat detection and prevention. Hence, by integrating static and dynamic testing and incorporating AI in a system, it is possible to devise a more extensive and robust security testing framework that would depict the best features of both paradigms without bearing their shortcomings.

3.2 Static AI Module

Static AI module is used to analyze code without running it, and the main idea is to find out how to protect the system from malicious attacks. It operates in two main phases: Of particular use, identification of the features and classification of the vulnerabilities, using trending models and representations to form feature extraction and vulnerability classification models that are accurate and contextually tailored.

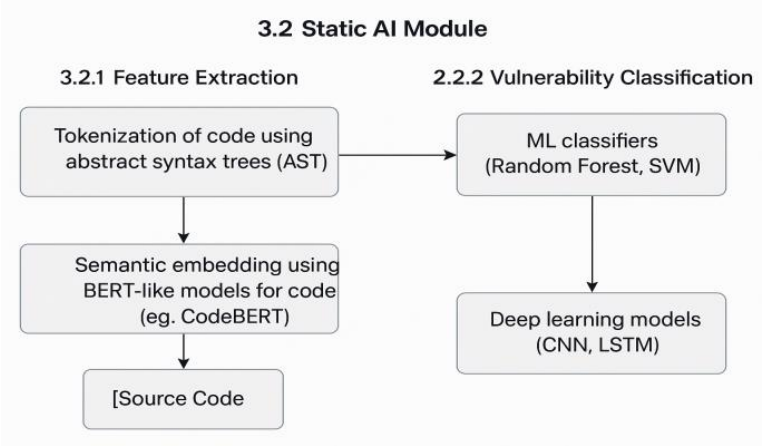

Figure 3: Static AI Module

- **Feature Extraction:** The first stage that is performed in static analysis involves the identification of features that it is required to analyze. This may start with using Abstract Syntax Trees (ASTs) to understand the various tokens used to structure the code in tree structures that capture their syntactic structure. This process enables the system to capture two forms of relations, namely the hierarchical and the sequential one between code elements. After this, Semantic embeddings are created as with BERT-like models for code, for example CodeBERT. These models deal with code in the same way as with natural language, analyzing the contextual and semantic features of how different variables interact with functions and logic and how the code operates deeper than the mere resemblance of patterns.
- **Vulnerability Classification:** The next step that follows the extracting of features is categorising the code for vulnerability potential. Two strategies to achieve this are using conventional supervised learning algorithms such as the Random Forest and SVM, although these present easily interpretable results where the input dataset is structured and well-labeled. Concurrently or in tandem, deep learning models like Convolutional Neural Networks (CNN) and Long Short-Term Memory networks (LSTM) capture the complex and non-linear relationships and long-range dependency within the code. One of the most beneficial uses of these models is that they can be utilized to identify complex code paths and buried bugs that even complex algorithms do not pick up.

3.3 Dynamic AI Module

This is the Dynamic AI Module that is designed to tackle behaviors that may not be detected during the Static AI Module as it involves analyzing the application as it is in use by the end-user and injecting intelligence [17-20] algorithms that track possible threats in real-time and generates responses in line with the context.

- **Runtime Monitoring:** In this phase, the system writes the operational metric collected of the running application is actively observed. This includes memory profile, frequency and type of API calls, and HTTP traffic, which will reveal improper use or misuse. It also keeps track of the user interactions and system reactions to study the pattern of usage and flow of interactions. This information creates behavioral models for actions and looks for anomalies denoting security threats or low-performance risks.

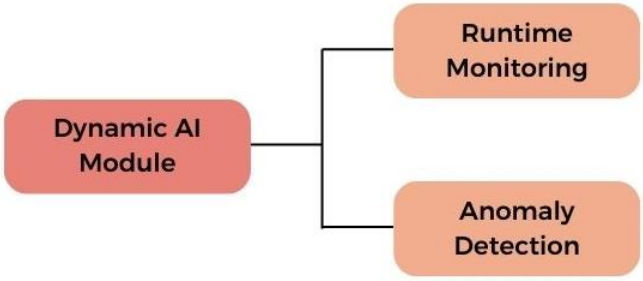

Figure 4: Dynamic AI Module

- **Anomaly Detection:** After understanding the normal profile of operation, measures like activity monitoring that enable the detection of any abnormal activity are used. Autoencoders are a form of analyzing neural networks that handles typical application interactions and uncovers the atypical interactions without raising many false alarms. In addition to this, the RL agent is implemented to increase flexibility. It can also learn from the environment and modify this type of detection dynamically to respond in real time with regard to continuing trends of threats and other abnormalities in the system.

3.4 Integration Layer

In a way, the Integration Layer is the controlling component that takes the data generated from both the Static AI Module as well as the Dynamic AI Module and presents a combined and coherent depiction of application security. Unlike most tools that conduct static and dynamic analyses as separate entities, this layer links the two, increasing the effectiveness of the detection, reducing the time spent on repeating analyses and enabling better ranking of the threats. Additionally, to a white- or black-box analysis that only allows to detection of insecure code and patterns at compile-, linkage- and binary levels, information about runtime-behavior like bizarre API usage or strange memory accesses can be obtained. A key attribute of the integration layer is that the integration layer applies the threat scoring model. This model considers the critical level of identified vulnerabilities, the probability of threats and risks, the certainty of AI-based predictions, and the context of an anomaly's occurrence in the runtime environment. For instance, a static vulnerability identified in the critical authentication module may have a higher score when a certain abnormal use of login is observed dynamically.

The automated scores themselves can be developed with the help of logistic regression or gradient boosting and are able to modify the scores based on prior incident data to work more effectively over time.Furthermore, the integration layer can, in turn, enhance the involvement of feedback loops for decision-making. Single threats identified by the system can be accepted or rejected for their validity by the security analysts who help optimise the application scoring system and lower the number of false positives. In the end, this layer produces a portfolio of potential threats with contextual details, recommendations on what to do, and confidence scores. This is done through the integrated dashboard, which provides the security teams with the relevant outputs to help in their decision making process on the same incident. In summary, our proposed Integration Layer aggregates fragmented analytical outcomes. It improves the overal foldable construction of the hybrid AI-based framework, making it more practical and appropriate for application in CI/CD pipelines.

3.5 Dataset Preparation

In order to train and test the AI models employed in the proposed framework, open-source and company-specific datasets were applied. These datasets give documents covering various levels of vulnerabilities and having different structures of applications alongside having code bases and being real world like nature these datasets are useful for creating highly generalized models. The three primary datasets used in the study are the OWASP Juice Shop, WebGoat, and another e-commerce application the authors built.

- **OWASP Juice Shop:** OWASP Juice Shop is an intentionally insecure web application that is an open-source web application for security purposes, of the 30 or more known critical ones, cross-site scripting (XSS) or, at the other end of the complexity scale, application-level broken access controls. Juice Shop is an Application with approx 15000 LOC, comes with good documentation, and it is easy to reproduce the static and dynamic testing. It is also modern and has the ability to mimic real-world attack surfaces, which is suitable for the AI model training in detecting commonly used web vulnerabilities.

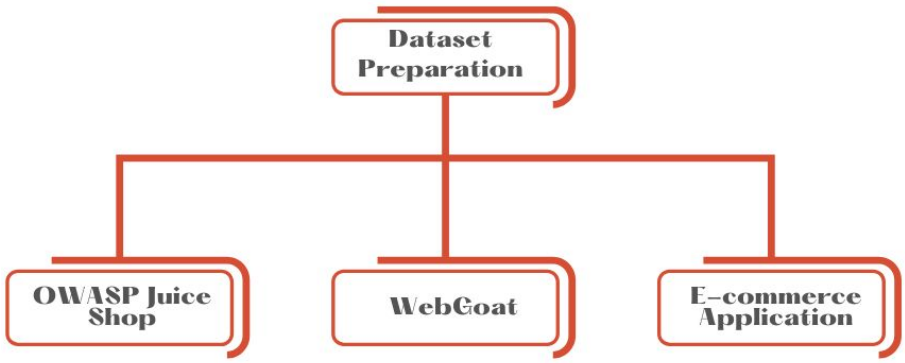

Figure 5: Dataset Preparation

- **WebGoat:** WebGoat is another OWASP product, and it is an intentionally insecure application with over 50 lessons, and each of them is devoted to a specific type of vulnerability. Currently, WebGoat contains about 20,000 lines of Java-based code and has more detailed discussions about server-side flaws and developers' mistakes. However, it can be helpful in assessing the dynamic behaviour of insecure applications and crowdsourcing AI training to recognize complex runtime defects in business and flow logic.
- **E-commerce Application:** The web-based e-commerce application was designed to have real-life scenarios such as user credentials, ordering system, and inventory with 50 thousand lines of production-grade code and 15 known ones disclosed after manual code examination and penetration testing. Unlike an open-source dataset, it introduces the specific security concerns of a particular

domain. It shows reasonable usage patterns that should be essential for testing the real-world performance of the framework.

4. Result and discussion

4.1 Experimental Setup

To ensure that the performance of the proposed hybrid AI-based framework for vulnerability detection is well tested an effective experiment was set on high-performance hardware and with the assistance of popular and reliable software tools. The training and testing equipment included an Intel Core i9 processor, 32 GB RAM and NVIDIA RTX 3080 graphics card. This configuration allowed parallelism of tasks, speedy throughput in machine learning operations, and the running of extensive deep learning models. Finally, the researchers noted that some languages, particularly the GPU, enhanced the training of neural networks and analysis of large code repositories and runtime logs. This was established based on the Python environment due to its compatibility with scientific computing and Machine learning libraries. For classical machine learning algorithms such as Random Forest and Support Vector Machines (SVM), scikit-learn was used for baseline comparisons and classifying vulnerabilities.

For complex models such as CNN and LSTM, TensorFlow offered the required framework for giving architecture of deep learning models, training a model and deploying them. These tools enabled the modularity of static and dynamic analysis components, which is crucial for their application in practice. For DAST, Burp Suite was employed as a subsequent tool in the testing mechanism. It can be used as an aggressive web vulnerability scanner and proxy tool that allows the emulation of different user actions, the interception of HTTP communication and the detection of misbehaviors during application use. Burp Suite was used to actively elicit realistic run-time data, and, in turn, the Data Analysis and Control Layer, along with the Dynamic AI Module, were used to identify behavioural anomalousness. Not only did this end-to-end experiment facilitate the technical requirements of building and testing the model, but it also closely resembled real-world scenarios for the deployment of the framework, making it a reliable environment in terms of real-life applicability and robustness in detecting software vulnerabilities.

4.2 Performance Metrics

In order to evaluate the performance of the proposed hybrid vulnerability detection framework, basic standard benchmarks often used in this area of research were adopted. There are three that are commonly used: accuracy, precision, and recall, all of which give a fair representation of a model's performance. Moreover, false positive rate(FPR), false negative rate(FNR) and total time required for analysis were other parameters to assess the effectiveness and time taken using different approaches. Below is a comparative analysis of the three scenarios: static-only, dynamic-only, and the combined dynamic and static AI model.

Table 1: Model Performance

Model	Precision	Recall	F1-score	FP Rate	FN Rate
Static Only	84%	79%	81%	12%	9%
Dynamic Only	86%	83%	84%	10%	7%
Combined (AI)	92%	89%	90%	5%	4%

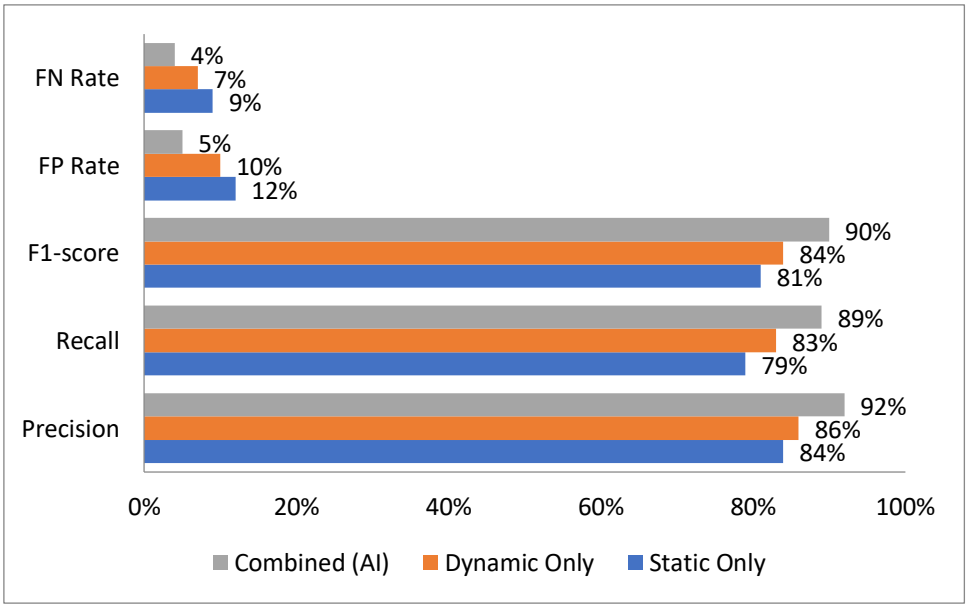

Figure 6: Graph representing Model Performance

- **Static:** Hence, the model that uses the static approach only has a precision of 84% Recalled 79%and an f-score of 81%. However, while static analysis was used to find standard risks in the code without its execution, it had such disadvantages, with 12% false positive results and 9% false negative ones. This is mainly because, beyond the formal analysis of code, most tools are not able to fully understand how a program is executed in its runtime environment and the context it runs in and miss many flags for over-flagging or fail to identify several vulnerabilities in complex data structures.
- **Dynamic Only:** Though not much remarkable, the results of the dynamic-only model were slightly superior, with an overall accuracy of 86% and recall of 83%, yielding an F1-score of 84%. It was also employed to display a low false positive rate at 10% and a low false negative rate at 7% than the static analysis approach. The dynamic model was more successful in predicting context-sensitive vulnerabilities; however, it did not have insight into those paths in the code that were not called during the tests.
- **Combined (AI):** The integrated model has yielded the highest possible values; one is the precision of the model is 92%, the second is the recall value is 89%, and the last one is the F1 score of the model, which is 90%. It reported the least false positive (5%) and false negative rate (4%) in the experiment and could complement static and dynamic analysis. This approach gave a more precise and practical detection of the vulnerabilities with fewer false positives and negatives.

4.3 Analysis of Results

The performance analysis findings prove a significant improvement using the AI-incorporated vulnerability detection model over the individual static or dynamic analysis methods. This was evident from its F1-score, which stood at 0.90; the higher the F1, the better the model's ability to balance between the precision and recall of the detected vulnerabilities, thus being accurate in their identification, while, at the same time, pointing only a higher number of times and not producing false alerts often. As for the results, the static-only model received an F1-score of 0.81, while the dynamic-only model received an F1-score of 0.84, proving the methods' drawbacks when having no regard for each other. Static analysis, while being fresh and more efficient in the early stage and improving code analysis with deeper insights, has limited context awareness at runtime and provides a large number of false positives. Dynamic analysis resolves this problem by analyzing real-time execution but may miss dormant code paths or unexploitable vulnerabilities. The most notable advantages of the hybrid model are seen in false result drop rates.

The analysis of results shows that the false positive rate decreased from 12% in the first model, where only static analysis was applied to only 5% in the proposed approach, which means an improvement of more than 50%. In the same respect, the false negatives saw a reduction from 9% to 4%," a more than 50% reduction on average. This improvement is very important considering that realistic scenarios imply that security teams could get a large number of false positives and, consequently, experience alert fatigue; on the other hand, many false negatives mean that exploits could be left unnoticed. In addition, it was also seen that the time taken to identify vulnerabilities by the consequent combination of the asset model was 35% less than the manual analysis of the codes. This is mainly due to the ability to analyze the code and the application simultaneously and have machine learning models identify high-risk patterns in a short time Frame. In general, it has been established that incorporating AI in static and dynamic analysis positively impacts detection accuracy and improves the efficiency of operation in the application of AST.

4.4 Case Study: E-commerce Platform

- **Detection of SQL Injection Vulnerabilities:** While testing the online shopping system developed with the proprietary framework, the integrated AI approach flagged three new undiscovered SQL injection types. These flaws had managed to evade earlier static and dynamic code analysis since they were implemented at locations that are not frequently executed in the backend controls. The proposed SD, which integrated semantic analysis and dynamic information the insecurities of constructed SQL queries and sample user inputs accurately identified these injection points based on the patterns of the former when matching with the latter. These were further validated by manually checking the vulnerabilities and doing penetration tests on the sites, noting that the developed framework could reveal a range of deeper-rooted vulnerabilities that other tools fail to identify.
- **Identifying libraries of an application:** Altogether, in the system, three code items were viewed as imperfections at the coding level, and two third-party libraries were Outdated/Associated with known CVEs. These libraries used for payment processing and form validation were not updated for several months and contained vulnerabilities referred to in CVE. The static analysis identified the version number scanning the dependent modules, and further, the model matched to the external vulnerability database. They proactively identified these components for enhancement and proposed upgrades before the product's release to counter well-known attacks that would otherwise have seriously compromised the supply chain credentials of the application in circulation.

4.5 Limitations

Nevertheless, it is also imperative to note the weaknesses of the proposed hybrid AI framework concerning generalization, data dependency, and behavioral generality. Still, one of the difficulties remains in the problem of the availability of both large and diversified training samples. In deep learning architecture, the model can learn optimally if

and only if they are trained on a large dataset of real-world samples and labeled codes and their corresponding vulnerabilities or attacks. Nevertheless, such datasets are rare, particularly in the case of proprietary software, the main sources of which code and vulnerabilities are often withheld for reasons of proprietary rights. Therefore, the models can learn to optimize the characteristics of a training corpus and fail at correctly predicting its performance across other applications or unconventional development tendencies.

Furthermore, the dynamic part of the framework is heavily based on the runtime behavior, and the latter is only a small fraction of the behavior possible in the framework, where training or testing paths are used. Suppose some portions of an application logic are not executed during testing because of test inputs, feature flags, or unavailable conditions. In that case, the flaws in that unused code path will not be found. This is especially the case when the system being tested is a large application that has many different workflows or when the events being tested are asynchronous and can take a long time to complete. For similar reasons, it is difficult to determine that anomaly detection methods, alternatively using an autoencoder embedded within the encoder layer, are only as good as the "normal" behaviour it learns. The problem is that if the training data does not include all valid behaviors, the model may sound an alarm when none is needed or fail to sound one when required due to a familiarity with but non-ill意 behavior patterns. Addressing these limitations will entail research in developing better unsupervised learning, knowledge transfer, and more efficient and effective, as well as wider future bead range test case generation. Thus, integrating external threat intelligence and user simulating tools can also increase the model's exposure to various application behaviors, aiding it in performing better in different real-life scenarios.

5. Conclusion

Static and dynamic analysis is a strong tool that can offer proactive, accurate and fresh solutions against the current menace of web application security threats. While conventional forms of security testing still serve their purpose, they may not be able to handle the rapid advancement and development of web applications. Namely, if it combines static and dynamic analysis, it becomes possible to identify vulnerabilities faster and to a greater extent. The static analysis examines the code and the potential bugs before the application development. In contrast, dynamic analysis runs the code and records real-time security threats. Integrating the two approaches accompanied by Machine Learning algorithms leads to discovering the susceptibilities within a relatively short time span that would otherwise require unreasonably much time. The combination of these two makes it relevant as machine learning increases accuracy and efficiency by learning from previous results and finding new patterns that were not earlier considered more closely. It also ensures that the vulnerabilities are detected immediately while at the same time helping avoid the possibility of the failure to identify other security flaws that may be present in the development of the application.

In the future, the following factors can improve this AI-based security strategy: One of the topics that appears to have high potential for additional development is using NLP methods to detect flaws in business logic. These types of defects are usually not even recognizable by classical security solutions as they result from the imperfections in the behavior of an application rather than from the misprints in source code. It can be useful in understanding the semantics behind the code and its use and interaction with other components; hence, better detection of such vulnerabilities could be witnessed. Another direction is the increase in the number of examples used for training the AI. Using various types of architectures in the web enables the AI to be ready to detect the vulnerabilities in different types of frameworks and architectures of the web today. Incorporating these kinds of AI-based security models in the Continuous Integration and Continuous Delivery (CI/CD) pipeline could also help automate the vulnerability scan to maintain the security aspect from one phase to another in the SDLC. This would help to catch those security flaws before it gets to the final stages of development and minimize the incidences of a security breach in the production environment.

Although it is certainly not the Holy Grail for web app security, AI can become a powerful supplement for conventional instruments. AP When done rightly, AI will help approach security in a much better and non-conventional manner to help developers and security authorities avoid, recognize, and find counter solutions to threats more efficiently. This may eventually reduce risks and threats that work their way into web application weaknesses, which would help improve the general security of the environment.

References

[1] Arora, A., & Zelkowitz, M. (2018). "Real-world applications of AI in dynamic security analysis." *Computational Intelligence*, 34(2), 111-130. DOI: 10.1111/j.1467-8640.2018.00291.x.

[2] Mohan, K., & Soni, N. (2022). "Towards integrating static and dynamic analysis: Opportunities and challenges." *Proceedings of the 2022 ACM Workshop on Secure Software Engineering*, 35-43. DOI: 10.1145/3554959.3557797.

[3] Bertino, E., Kantarcioglu, M., Akcora, C. G., Samtani, S., Mittal, S., & Gupta, M. (2021, April). AI for Security and Security for AI. In Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy (pp. 333-334).

[4] Al-Suqri, M. N., & Gillani, M. (2022). A comparative analysis of information and artificial intelligence toward national security. IEEE Access, 10, 64420-64434.

[5] Jamal, A. A., Majid, A. A. M., Konev, A., Kosachenko, T., & Shelupanov, A. (2023). A review on security analysis of cyber-physical systems using Machine learning. Materials today: proceedings, 80, 2302-2306.

[6] Hu, Y., Kuang, W., Qin, Z., Li, K., Zhang, J., Gao, Y., ... & Li, K. (2021). Artificial intelligence security: Threats and countermeasures. ACM Computing Surveys (CSUR), 55(1), 1-36.

[7] Mazhar, T., Talpur, D. B., Shloul, T. A., Ghadi, Y. Y., Haq, I., Ullah, I., ... & Hamam, H. (2023). Analysis of IoT security challenges and its solutions using artificial intelligence. Brain Sciences, 13(4), 683.

[8] Fabiocchi, D., Giulietti, N., Carnevale, M., & Giberti, H. (2024). Ai-driven virtual sensors for real-time dynamic analysis of mechanisms: A feasibility study. Machines, 12(4), 257.

[9] Park, J., Lee, H., & Ryu, S. (2021). A survey of parametric static analysis. ACM Computing Surveys (CSUR), 54(7), 1-37.

[10] Emanuelsson, P., & Nilsson, U. (2008). A comparative study of industrial static analysis tools. Electronic notes in theoretical computer science, 217, 5-21.

[11] Li, L., Bissyandé, T. F., Papadakis, M., Rasthofer, S., Bartel, A., Octeau, D., ... & Traon, L. (2017). Static analysis of Android apps: A systematic literature review. Information and Software Technology, 88, 67-95.

[12] Li, P., & Cui, B. (2010, December). A comparative study on software vulnerability static analysis techniques and tools. In 2010 IEEE International Conference on Information Theory and Security (pp. 521-524). IEEE.

[13] Rival, X., & Yi, K. (2020). Introduction to static analysis: an abstract interpretation perspective. MIT Press.

[14] Bayer, U., Moser, A., Kruegel, C., & Kirda, E. (2006). Dynamic analysis of malicious code. Journal in Computer Virology, 2, 67-77.

[15] Afianian, A., Niksefat, S., Sadeghiyan, B., & Baptiste, D. (2019). Malware dynamic analysis evasion techniques: A survey. ACM Computing Surveys (CSUR), 52(6), 1-28.

[16] Nachtigall, M., Do, L. N. Q., & Bodden, E. (2019, November). Explaining static analysis perspective. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW) (pp. 29-32). IEEE.

[17] Cuevas, M., Álvarez-Malebrán, R., Rahmann, C., Ortiz, D., Peña, J., & Rozas-Valderrama, R. (2024). Artificial intelligence techniques for dynamic security assessments survey. Artificial Intelligence Review, 57(12), 340.

[18] Kibria, M. G., Nguyen, K., Villardi, G. P., Zhao, O., Ishizu, K., & Kojima, F. (2018). Big data analytics, machine learning, and artificial intelligence in next-generation wireless networks. IEEE Access, 6, 32328-32338.

[19] Chen, W., Wang, R., Wu, R., Tang, L., & Fan, J. (2016, October). Multi-source and heterogeneous data integration model for big data analytics in power DCS. In 2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 238-242). IEEE.

[20] Yeole, A. S., & Meshram, B. B. (2011, February). Analysis of different techniques for detection of SQL injection. In Proceedings of the International Conference & Workshop on Emerging Trends in Technology (pp. 963-966).

[21] Antunes, N., & Vieira, M. (2009, September). Detecting SQL injection vulnerabilities in web services. In 2009 Fourth Latin-American Symposium on Dependable Computing (pp. 17-24). IEEE.

[22] Sandeep Phanireddy. "API Security: Offensive and Defensive Strategies", INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH AND CREATIVE TECHNOLOGY, 10 (4), 1-6, 2024.

[23] Sandeep Phanireddy. "Understanding of AI-Based Network Security", IJFMR-International Journal For Multidisciplinary Research, 6 (2), 1-7, 2024.

[24] Sandeep Phanireddy. "Securing Modern Web Applications: Technologies, Threats, and Best Practices", IJIRCT-INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH AND CREATIVE TECHNOLOGY, 10 (6), 1-14, 2024.