Detection and prediction of the vulnerabilities in software systems based on behavioral analysis with machine learning

Yevheniy Sierhieiev[1,*,†], Vadym Paiuk[1,†], Andrii Nicheporuk[1,†], Andrzej Kwiecien[2,†], Oleksandr Huralnyk [1,†]

[1] *Khmelnytskyi National University, Institutska str., 11, Khmelnytskyi, 29016, Ukraine*
[2] *Silesian University of Technology, Akademicka str., 2A, Gliwice, Poland*

Abstract

This study introduces Behavioral Analysis with Machine Learning (BAML), a novel approach designed to enhance cybersecurity by utilizing machine learning algorithms to detect and predict vulnerabilities in software systems based on behavioral data.

BAML integrates both supervised and unsupervised learning techniques to analyze extensive datasets comprising system calls, network traffic, and user interactions. This method continuously monitors software operations, comparing observed behaviors against a machine-learned model to identify deviations that signal potential vulnerabilities.

The effectiveness of BAML was assessed through a series of controlled experimental studies comparing its performance against traditional security testing methods such as Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Interactive Application Security Testing (IAST). BAML demonstrated superior accuracy with a true positive rate of 94%, the lowest false positive rate at 11%, and the highest code coverage of 93%. It also excelled in zero-day vulnerability detection and complex dependency analysis, showcasing its ability to adapt and respond to emerging threats dynamically.

BAML offers significant advancements in the detection and prevention of software vulnerabilities. Its ability to learn from continuous data streams and adapt to new threats in real-time positions it as an essential tool for modern cybersecurity strategies, aligning well with Agile and DevOps practices. This proactive approach not only improves security but also reduces the costs and efforts associated with traditional reactive security measures.

Keywords

vulnerabilies, cyber security, threat detection, cyber defense, SATS, vulnerable detection,

1. Introduction

In the fast-developing field of cybersecurity, it is crucial to continuously develop and implement strong security measures to protect digital infrastructure. Traditional security methods often

ICyberPhyS-2024: 1st International Workshop on Intelligent & CyberPhysical Systems, June 28, 2024, Khmelnytskyi, Ukraine

* Corresponding author.
† These authors contributed equally.
ysierhieiev@gmail.com (Ye. Sierhieiev); vadympaiuk@gmail.com (V. Paiuk); andrey.nicheporuk@gmail.com (A. Nicheporuk); andrzej.kwiecien@polsl.pl (A. Kwiecien); mailto:gurualexua@gmail.com (O. Huralnyk)
0009-0008-9877-9863 (Ye. Sierhieiev); 0000-0002-7253-893X (V. Paiuk); 0000-0002-7230-9475 (A. Nicheporuk); 0000-0003-1447-3303 (A. Kwiecien); 0009-0009-1175-8726 (O. Huralnyk)

struggle to keep up with the advanced tactics used by modern cyber threats. This challenge is further complicated by the dynamic nature of digital interactions and the increasing complexity of software systems. To effectively address these vulnerabilities, it is essential to go beyond conventional approaches and incorporate more adaptive technologies such as machine learning.

This article discusses how integrating behavioral analysis with machine learning techniques can enhance the detection and prevention of security vulnerabilities. Behavioral Analysis with Machine Learning (BAML) represents a new approach to cybersecurity, focusing on dynamic and proactive threat detection. BAML uses patterns derived from normal and abnormal software behaviors to predict and identify potential vulnerabilities before they can be exploited. This technique adapts to evolving threats in real-time, continuously learning from new data. It offers a robust defense mechanism that is both scalable and efficient.

BAML's foundation is based on collecting and analyzing behavioral data, including system calls, network traffic, and user interactions. This data is processed using advanced machine learning algorithms to identify deviations from expected behavior, which indicate potential security threats. By using techniques such as supervised and unsupervised learning, along with more complex models like deep learning, BAML has a better understanding of software behaviors. This enables the early detection of sophisticated cyber threats that may bypass traditional security measures.

Relevant to this discussion, the research by Pomorova et al. on "A Technique for the Botnet Detection Based on DNS-Traffic Analysis"[1] and by Lysenko et al. on "DNS-based Anti-evasion Technique for Botnets Detection"[2] highlight the significance of DNS traffic analysis in identifying anomalies associated with botnets. These studies provide crucial context and support the necessity of incorporating DNS analysis into BAML, enhancing its ability to detect similar complex security threats.

In the upcoming sections, we will explore the specifics of BAML, including its operational framework, the integration of various machine learning models, and how it compares to traditional security testing methods such as Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST). Through a series of experimental studies, we will demonstrate the practical applications and advantages of implementing BAML in different software environments, highlighting its potential to revolutionize cybersecurity practices.

2. Classification of security testing in software

Security testing in software is a crucial process aimed at identifying vulnerabilities that attackers could exploit. This type of testing has evolved from manual code reviews and security audits to include a variety of automated tools that enhance efficiency and coverage. Security testing needs to be integrated throughout the software development lifecycle to ensure that vulnerabilities can be identified and mitigated from the earliest stages of development. Methods such as Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are employed to analyze both the source code and the running application, respectively. Despite the advancements in automated and dynamic testing techniques, the increasing complexity of software and rapid development cycles continue to pose significant challenges. However, the integration of machine learning techniques is seen as a promising direction for future advancements in security testing, potentially allowing for the prediction and detection of complex vulnerabilities before they become critical threats[3, 4, 5].

In this article, we aim to examine various methods of vulnerability analysis:

- Static Application Security Testing (SAST): SAST is an essential method of security testing where the source code, bytecode, or binary code is analyzed for vulnerabilities without executing the program. This type of testing is performed at the earliest stages of software development, allowing developers to identify and fix security issues before the software is run. SAST tools are designed to be integrated into the development environment, providing immediate feedback to developers as they code, which helps in ensuring that security vulnerabilities are addressed as soon as they are introduced[6, 7].
- Dynamic Application Security Testing (DAST): DAST tools are used to detect conditions that indicate a security vulnerability in an application while it is running. This method of testing interacts with an application from the outside, mimicking an attacker's approach to understand the application's behavior during execution. DAST is effective in identifying runtime issues such as session management problems and data validation issues, which are not detectable by SAST[8, 9].
- Interactive Application Security Testing (IAST): Combining aspects of both SAST and DAST, IAST tools work from within an application to monitor its behavior and the data it processes in real-time. IAST tools are capable of identifying security vulnerabilities while the application is under testing, offering a more comprehensive analysis by observing the application during interaction and from within. This method provides the advantages of both static and dynamic approaches, leading to fewer false positives and more accurate detection of complex vulnerabilities[10, 11].
- Runtime Application Self-Protection (RASP): RASP technology is integrated or linked with an application's runtime environment and actively monitors its behavior to detect and block potential attacks in real-time. Unlike SAST and DAST that are used during testing phases, RASP provides protection while the application is in production, offering an immediate response to security threats without human intervention. This method shifts some of the security responsibility from developers to the application itself, enabling more robust security defenses during an application's operational phase[12, 13].

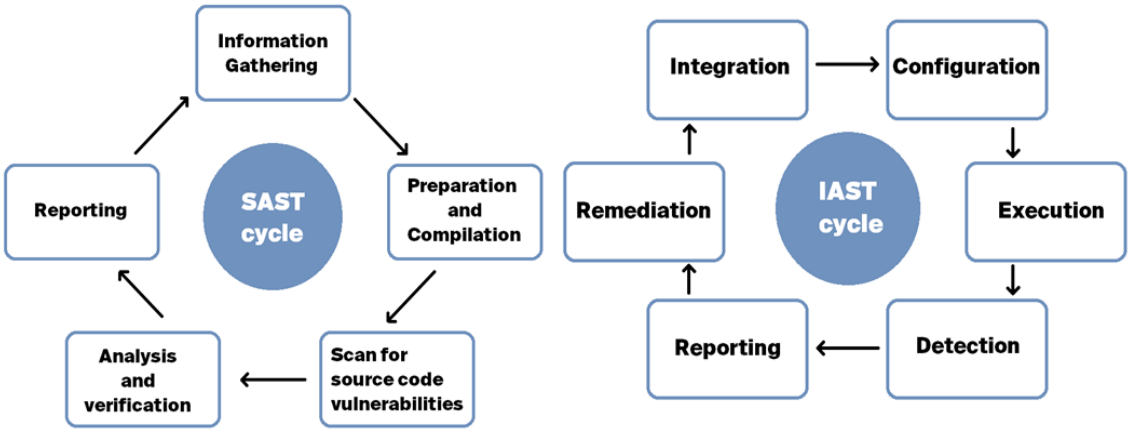

Figure 1: SAST and IAST cycle.

These security testing technologies reflect the diverse approaches required to effectively address the myriad of security challenges in today's software development environments. Each type offers unique benefits and plays a crucial role in a comprehensive software security strategy, ensuring applications are robust against both known and emerging security threats[13, 14].

Security testing in software, encompassing methodologies like SAST, DAST, IAST, and RASP, has become indispensable in the intricate landscape of software development, marked by an ever-expanding array of platforms and environments, including cloud services, mobile applications, and extensive enterprise systems. The integration of these security practices within Agile and DevOps workflows has revolutionized the way vulnerabilities are addressed, embedding them within continuous integration/continuous deployment (CI/CD) pipelines to ensure timely detection and remediation.[15,16] This integration not only mitigates the risk of security breaches but also curtails the costs associated with late-stage corrections. However, the implementation of automated testing tools, while scaling security efforts, presents challenges like alert fatigue and the potential disruption of established development workflows, necessitating adaptations in team dynamics and methodologies.

The advancement of security testing is also seeing the introduction of artificial intelligence and machine learning, which leverage historical data and behavioral patterns to predict and preemptively counter threats, thereby enhancing the proactive capabilities of security measures. Moreover, the growing stringency of compliance and regulatory frameworks demands rigorous adherence to standards such as GDPR, HIPAA, and PCI-DSS, further embedding tools like RASP not just for threat mitigation but also for ensuring compliance[17]. Embracing a comprehensive approach to security, continually refining and evolving strategies to meet new challenges, is critical for organizations aiming to protect their software assets against both existing and emergent threats, thus weaving robust security protocols into the very fabric of software development practices.

Recent research underscores the effectiveness of these advanced methodologies. For instance, Markowsky et al. have developed a novel technique for the detection of metamorphic viruses based on their obfuscation features, significantly improving the detection of sophisticated malware that traditional methods might miss. This technique highlights the potential of using detailed analysis of code changes and behavior to detect threats that evolve to bypass conventional detection methods.

Additionally, the work of Lysenko, Savenko, and Bobrovnikova on the application of Semi-Supervised Fuzzy c-Means Clustering for DDoS botnet detection illustrates how machine learning can be applied to classify network traffic and identify malicious patterns effectively[18]. This approach not only enhances the accuracy of botnet detection but also adapts to new and evolving botnet behaviors that might not yet be fully understood or documented.

Furthermore, the study on DNS Tunneling Botnets by Savenko et al. presents a technique that detects covert communication channels often used by attackers to exfiltrate data or command and control botnets. By analyzing DNS requests and responses for anomalies, this method provides an additional layer of security to identify and block these sophisticated threats.

Embracing a comprehensive approach to security, continually refining and evolving strategies to meet new challenges, is critical for organizations aiming to protect their software assets against both existing and emergent threats. This integration of advanced detection

techniques and compliance measures is becoming indispensable in the intricate landscape of software development, marked by an ever-expanding array of platforms and environments, including cloud services, mobile applications, and extensive enterprise systems

3. Related works

Seyed Mohammad Ghaffarian and Hamid Reza Shahriari's 2017 survey, published in the ACM Computing Surveys, rigorously analyzes the application of machine learning and data mining methods to software vulnerability detection. This extensive survey presents a critical discussion on the use of various machine learning approaches, such as neural networks and random forests, highlighting the high-quality outcomes particularly achieved by random forest algorithms in identifying software vulnerabilities. Furthermore, the paper details the methodology of code similarity analysis, which decomposes software into fragments for comparison using abstract representations like tokens, trees, and graphs, aiding in the detection of similar vulnerabilities[19].

The study introduces the concept of "vulnerability extrapolation," a process for uncovering previously unknown vulnerabilities by detecting patterns in existing security issues. It also outlines a set of metrics for classes and methods that serve as features in machine learning models to predict vulnerabilities. This approach not only supports the identification of vulnerabilities but also emphasizes the role of automated deep learning and machine learning analysis methods in enhancing the effectiveness of vulnerability detection. Ghaffarian and Shahriari's work contributes significantly to the ongoing evolution of security testing methodologies, offering a comprehensive framework that could potentially set new directions for future research in software security analysis.

Valentina Lenarduzzi, Fabiano Pecorelli, and Nyyti Saarimäki focuses on the evaluation and critical analysis of six static analysis tools to improve software security and integrity. The authors highlight the importance of integrating these tools within the software development lifecycle to detect and mitigate potential security vulnerabilities early in the development process. The study underscores the significant role that static analysis tools play in enhancing software security, by providing empirical evidence on their effectiveness in identifying common vulnerabilities and coding errors before software deployment[20]. This research is pivotal as it not only reinforces the necessity of static analysis in security testing but also offers a comparative analysis that aids developers in selecting the most effective tools for their specific needs, making it a crucial reference for those involved in developing secure software systems.

Also Mateo Tudela, F. et al., and by Pupo, A. L. S. et al. both discuss advancements in static and dynamic testing methods for software security, but they approach the integration of these methods with distinct emphasis and methodologies. They focus on combining static, dynamic, and interactive approaches to enhance software testing processes[21]. Their approach is holistic, aiming to cover a broad spectrum of vulnerabilities by leveraging the strengths of each testing method. They argue that the fusion of these methods provides a more robust and comprehensive security evaluation, capable of detecting more subtle and complex vulnerabilities that might be missed when these methods are used in isolation.

On the other hand, Pupo, A. L. S. et al. concentrate specifically on the integration of static security testing (SST) with software development practices to emphasize early detection of vulnerabilities. Their research highlights the effectiveness of SST in the initial stages of

development, reducing the cost and effort associated with later-stage corrections. This study particularly stresses on real-time feedback loops that incorporate SST findings directly into the development process, thus fostering a proactive approach to security. Both studies underscore the importance of integrating security testing into the development lifecycle but differ in their strategic application. Mateo Tudela, F. et al. advocate for a comprehensive combination of testing techniques throughout the development stages, while Pupo, A. L. S. et al. highlight the specific benefits of early-stage static testing. These differing approaches provide valuable insights into how security testing can be optimized to address different aspects of vulnerability detection and management within software development projects[22].

In their study, Koala, G., Bassolé, D., Tiendrébéogo, T., Sié, O. explore the effective use of software execution traces for enhancing vulnerability detection in software systems. They discuss how malicious attacks exploit vulnerabilities and emphasize the crucial role of execution traces in identifying these weak spots proactively. The research sheds light on how these traces provide detailed insights into the execution flow of applications, enabling the precise detection of anomalous behaviors and security weaknesses[23].

The research highlights the effectiveness of this method in enhancing the security of software systems by detecting vulnerabilities early in the development and deployment phases. This approach is particularly valuable as it allows developers and security analysts to intervene promptly, thus mitigating the risks associated with potential security breaches. The article makes a significant contribution to the field of cybersecurity by demonstrating how execution traces can be leveraged to bolster software security through proactive detection and resolution of vulnerabilities[24]. This technique serves as a powerful tool in the arsenal of software security testing, providing a dynamic method to safeguard applications from emerging threats.

In the research conducted by Amankwah, Richard, Kudjo, Patrick, and Yeboah, Samuel, the focus is on exploring different methods for detecting software vulnerabilities. The study delves into various techniques and tools employed in the identification and mitigation of security weaknesses in software systems. It emphasizes the continuous nature of software vulnerability, highlighting the necessity for ongoing detection and management strategies to safeguard against potential security breaches effectively.

Meanwhile, the work of Zhang, S., Caragea, D., and Ou, X. delves into an empirical analysis of software vulnerabilities using data from the National Vulnerability Database (NVD)[25, 26, 27]. This study emphasizes the critical nature of vulnerabilities in software systems and utilizes a comprehensive data-driven approach to understand the patterns and trends of software vulnerabilities over time. The analysis provides significant insights into the common characteristics of vulnerabilities and helps in refining the strategies for their detection and mitigation.

Both studies contribute valuable perspectives to the field of cybersecurity, with Amankwah and his colleagues focusing on the application and effectiveness of different vulnerability detection methods, and Zhang and his team providing a quantitative analysis of vulnerabilities to better understand their evolution and characteristics. These complementary approaches offer a broader understanding of how vulnerabilities can be detected, analyzed, and addressed in software systems.

The dissertation by Andersson, F., and Öberg, A. investigates predicting vulnerabilities in third-party open-source software (OSS) using data mining and machine learning techniques. Their study utilized data from GitHub repositories linked with vulnerabilities in the National

Vulnerability Database (NVD). They analyzed over 30,000 OSS package instances, finding patterns between GitHub features and reported vulnerabilities. The findings demonstrated a high prediction accuracy of 91.7%, with significant relationships between repository features like stars and forks and the prediction outcomes. This research contributes to enhancing digital system security by showing how machine learning can effectively forecast OSS vulnerabilities[28].

Complementing the insights provided by J. D. Pereira, N. Ivaki, and M. Vieira on buffer overflow vulnerabilities, and the advancements in web crawling technology by Wan, B., Xu, C., & Koo, J., the study by Pomorova et al. introduces a unique dimension by focusing on the detection of bots using polymorphic code. This research, detailed in their paper published in the Communications in Computer and Information Science, explores sophisticated techniques for identifying bots that dynamically alter their code to evade detection systems. This approach is crucial for preempting bot-based threats and enhances the overall strategies for cybersecurity alongside the preventive measures discussed in the previously mentioned studies. Together, these articles showcase a range of methods aimed at fortifying digital systems against diverse and evolving threats. [29,30, 31].

Building on this foundation, further research efforts continue to advance cybersecurity methodologies. Notably, additional studies by Pomorova et al. delve into metamorphic virus detection through modified emulators, expanding our understanding of adaptive malware challenges. Similarly, Kashtalian et al. explore robust multi-computer malware detection systems designed to handle metamorphic functionalities, offering a broader defense mechanism in cybersecurity. Additionally, Savenko, et al. introduce an innovative dynamic signature-based detection approach using API call tracing, significantly refining malware identification processes. These studies collectively emphasize the ongoing need for sophisticated, adaptable security solutions in response to complex cyber threats. [32, 33, 34]

4 Behavioral Analysis with Machine Learning (BAML)

After a thorough review of existing software vulnerability detection methods and understanding their strengths and weaknesses, we have developed an innovative method termed "Behavioral Analysis with Machine Learning" (BAML). This method enhances traditional techniques by incorporating a dynamic and real-time analysis of software behavior using advanced machine learning algorithms. BAML not only detects anomalies that may indicate potential vulnerabilities more effectively but also predicts potential security issues by learning from ongoing application behavior. Our method is designed to combat a wide range of cybersecurity threats, including malware, botnets, DDoS attacks, web application attacks like SQL injections and XSS, encrypted traffic analysis, internal threats, and data leaks, as well as advanced persistent threats (APT). By utilizing machine learning to analyze and predict unusual behavior patterns in software and network traffic, BAML provides a comprehensive tool for detecting both conventional and emerging cybersecurity threats, making it an effective solution in modern security strategies.

The foundational concept of BAML is to continuously monitor software operations and compare them against a machine-learned model of expected behavior. Deviations from this model are flagged as potential vulnerabilities. Here’s a step-by-step breakdown of how BAML operates:

1. The first step in deploying BAML involves extensive data collection to establish a baseline of normal software behavior. Advanced monitoring tools are used to gather data on system calls, network traffic, user interactions, and API usage. This comprehensive data set serves as the basis for training the machine learning model and is crucial for accurate anomaly detection.
2. In this step, a machine learning model is trained using the extensive behavioral data collected from the software. To effectively recognize and categorize both known behaviors and emerging vulnerabilities, the model integrates a combination of machine learning techniques:

- Supervised Learning: For known behaviors and vulnerabilities, supervised learning models such as Support Vector Machines (SVMs), Random Forests, and Gradient Boosting Machines (GBMs) are used. These models are trained on labeled datasets that include examples of normal and malicious activities to learn how to accurately classify and predict similar instances in the future.
- Unsupervised Learning: To identify new and unusual patterns that may indicate potential vulnerabilities, unsupervised learning methods like K-Means Clustering, Autoencoders, and Isolation Forests are employed. These methods analyze data without pre-labeled outcomes to detect anomalies and outliers in software behavior, which could signify a security threat.
- Deep Learning: For more complex pattern recognition tasks, such as detecting subtle anomalies in large-scale data, deep learning architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are particularly useful. These networks can process and learn from sequential or time-series data, making them ideal for monitoring continuous streams of behavioral data from software applications.
- Reinforcement Learning: Although less common in traditional vulnerability detection, reinforcement learning can be adapted to enhance decision-making processes within the model. It could potentially be employed to optimize the actions taken in response to detected anomalies, learning over time which responses are most effective in mitigating potential threats.

The model is regularly updated with new data to ensure that it adapts to the latest security threats and continues to reflect the current operational profile of the software. This ongoing training process helps to maintain the model's effectiveness and accuracy in real-time vulnerability detection.

3. Once trained, the BAML model is deployed to monitor the software in real-time. It continuously analyzes incoming behavioral data, comparing it against the baseline model to detect any significant deviations. These deviations, or anomalies, are flagged for further analysis, as they may indicate potential security threats.
4. Detected anomalies are assessed using a specific formula to calculate the probability of vulnerabilities. Equation 1.

$$P(v) = \sigma \cdot (\alpha \cdot f(B(x)) + \beta \cdot H(X) \quad (1)$$

5. Where $P(v)$ represents the probability of vulnerability presence, σ is a sigmoid function that normalizes the output to a probability range between 0 and 1, α and β are coefficients that balance the influence of behavioral analysis and historical data, $f(B(x))$ is a function evaluating behavioral data, and $H(x)$ integrates historical vulnerability data to refine predictions. Anomalies with high probability values are considered potential vulnerabilities and prioritized for immediate action.
6. The final step involves reporting the detected vulnerabilities. Automated alert systems notify security teams about potential vulnerabilities identified by BAML. Based on the assessed risk and potential impact, the security teams then initiate appropriate responses, which may include applying security patches, conducting further investigations, or undertaking full-scale security audits to mitigate the risks.

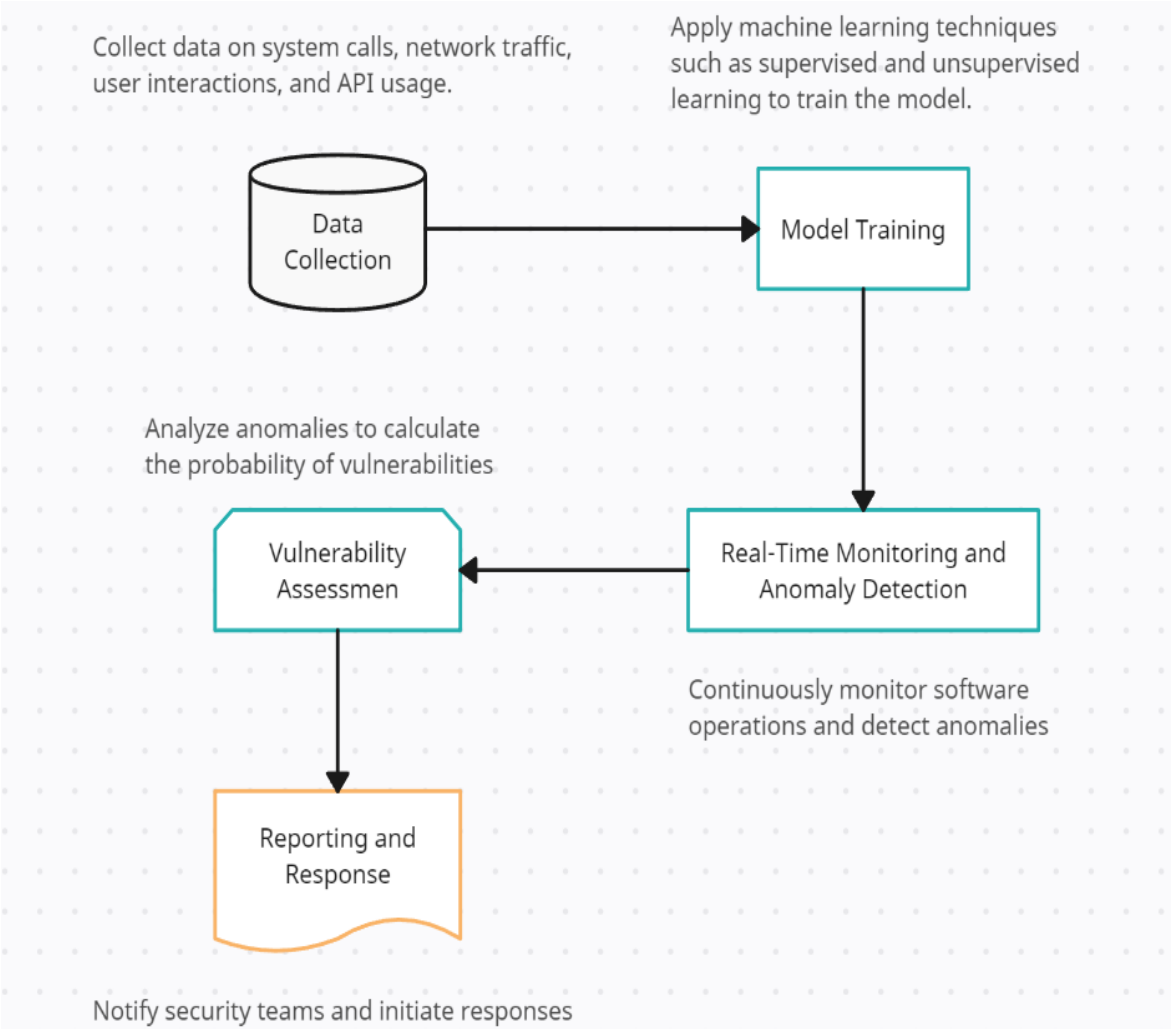

Figure 2: BAML process.

Behavioral Analysis with Machine Learning (BAML) offers a sophisticated and proactive tool for enhancing software security. By combining machine learning with dynamic behavioral analysis, BAML not only effectively identifies existing vulnerabilities but also provides

capabilities to anticipate and mitigate potential future threats. This method provides a comprehensive approach to safeguarding software systems in an increasingly complex digital environment, making it an essential component of contemporary cybersecurity strategies.

Behavioral Analysis with Machine Learning (BAML) inherently employs a variety of machine learning techniques to enhance software vulnerability detection. By diversifying and optimizing these techniques, BAML can be tailored to meet specific security needs more effectively, adapting to the nuances of different data types, anomaly patterns, and operational environments. Each machine learning approach contributes distinct strengths to the overall detection process, enabling BAML to achieve more nuanced analysis and robust detection capabilities.

Machine learning techniques such as Supervised Learning are foundational to BAML, enabling it to identify known vulnerabilities efficiently [35]. Models like Support Vector Machines (SVMs) and Random Forests are particularly effective in classification tasks, making them suitable for distinguishing between normal operations and potential security threats. These models excel in environments with well-labeled training data, allowing them to learn and predict based on historical patterns of vulnerabilities.

Unsupervised Learning plays a crucial role when dealing with unlabeled data, helping to uncover new and emerging threats. Techniques such as K-Means Clustering and Autoencoders are instrumental in detecting unusual patterns that do not match any known behavior, thus flagging them as potential anomalies. This is particularly valuable for identifying zero-day vulnerabilities and other novel threats that have not yet been cataloged.

Deep Learning further enhances BAML's ability to process and analyze complex data structures. With the capacity to handle large-scale and high-dimensional data, methods like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are adept at recognizing subtle anomalies over time. These models are excellent for continuous monitoring of network traffic and user activities, providing insights that more traditional machine learning models might miss.

Additionally, Reinforcement Learning can augment BAML by optimizing the decision-making processes regarding the response to detected threats. This approach adapts over time, learning which mitigation strategies are most effective in various scenarios, thus continuously improving the system's responsiveness and accuracy.

By incorporating these diverse machine learning techniques, BAML not only solidifies its capability to detect and react to existing threats but also enhances its predictive power. This strategic application of machine learning ensures that BAML remains effective as new security challenges emerge, maintaining high adaptability and precision in dynamic and complex digital environments.

In conclusion, the Behavioral Analysis with Machine Learning (BAML) method represents considerable progress in the field of cybersecurity, particularly in the domain of software vulnerability detection. This innovative approach leverages a comprehensive array of machine learning techniques to offer a more dynamic, adaptable, and proactive security solution.

BAML's core strength lies in its ability to lifelong learn and adapt to new threats. Unlike traditional security systems that rely on static databases of known vulnerabilities, BAML uses ongoing data collection and machine learning to constantly refine and update its understanding of what constitutes normal and anomalous behavior. This ability to adapt in real-time is crucial in today's rapidly changing threat landscape, where new vulnerabilities and sophisticated cyber

attacks emerge more frequently than ever before. The integration of various machine learning models, including supervised and unsupervised learning, deep learning, and reinforcement learning, allows BAML to be highly effective across different stages of the threat detection process. From identifying established patterns of attacks using supervised learning techniques to detecting subtle, novel threats with unsupervised and deep learning, BAML covers a broad spectrum of detection capabilities. Furthermore, reinforcement learning adds a strategic layer, enabling the system to make intelligent decisions about threat mitigation based on past interactions and outcomes. However, the implementation of BAML is not without challenges. The effectiveness of the system heavily depends on the quality, volume, and veracity of the data it processes. Ensuring the integrity and accuracy of data is paramount, as any anomalies in the data can lead to false positives or missed detections. Moreover, the complexity of configuring and maintaining such a sophisticated system requires significant expertise and resources, which may be a barrier for some organizations.

Looking forward, the potential for BAML to integrate with other emerging technologies could further enhance its capabilities. For example, the incorporation of artificial intelligence (AI) advancements could enable more nuanced analyses of complex behaviors and interactions within software environments. Additionally, the application of quantum computing might someday dramatically increase the processing power available for machine learning models, allowing for even faster and more accurate analyses. The ongoing development and enhancement of BAML will likely focus on improving its scalability and ease of integration. As it becomes capable of handling larger datasets more efficiently and seamlessly integrating into existing IT infrastructure, BAML could become an indispensable tool for a wide range of industries facing cybersecurity threats.

In sum, BAML not only enhances current capabilities in vulnerability detection but also sets the stage for future developments in cybersecurity practices. By pushing the boundaries of what machine learning can achieve in a security context, BAML offers hope for a more secure digital future, giving organizations the tools they need to defend against the ever-growing and evolving landscape of cyber threats.

5 Experimental studies

To validate the effectiveness of the Behavioral Analysis with Machine Learning (BAML) method, a series of experimental studies were conducted. These studies compared BAML against established vulnerability detection methods such as Static Analysis Security Testing (SAST), Dynamic Analysis Security Testing (DAST), and Interactive Application Security Testing (IAST). The experiments aimed to assess the detection accuracy, speed, cost-effectiveness, and ability to handle false positives.

The experiments were conducted using a controlled test environment that included various software applications. These applications were chosen to represent a range of use cases and included known vulnerabilities of different types and complexities, such as SQL injections, cross-site scripting (XSS), and buffer overflows.

For BAML, extensive behavioral data was collected during normal and malicious operations of the test applications. This data served as the basis for training the BAML model and for real-time analysis during the experiments.

Tools and Methods Used:

- SAST: Tools like SonarQube and Checkmarx were used to perform static code analysis.
- DAST: Tools such as OWASP ZAP and Burp Suite were employed to conduct dynamic testing on running applications.
- IAST: Tools like Contrast Security and Synopsys Seeker were utilized, combining elements of SAST and DAST for interactive testing

To evaluate the effectiveness of Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), Interactive Application Security Testing (IAST), and Behavioral Analysis with Machine Learning (BAML), each method was assessed based on several critical metrics.

True Positives (%) measures the accuracy in identifying actual vulnerabilities, while False Positives (%) assesses the rate at which non-threats are mistakenly flagged as vulnerabilities. Analysis Time (minutes) indicates the speed with which each method completes an assessment, critical in fast-paced development environments. Code Coverage (%) reflects the extent of the application code that the method can analyze. Zero-Day Vulnerability Detection evaluates each method's ability to identify previously unknown threats, crucial for cutting-edge security. Complex Dependency Analysis checks the capability of the methods to detect and analyze intricate dependencies that could affect security. Ease of Integration shows how smoothly each method can be incorporated into existing workflows, and Scalability indicates how well the method can handle increasing amounts of data or complexity as the organization grows. These parameters collectively provide a comprehensive overview of the performance of each security testing method, allowing for an informed choice based on specific operational needs and security requirements. The results of these assessments are summarized in Table 1.

Table 1
Results

Parameter	SAST	DAST	IAST	BAML
True Positives(%)	87%	89%	93%	94%
False Positives(%)	30%	24%	14%	11%
Analysis Time (minutes)	30	42	20	20
Code Coverage (%)	75%	67%	88%	93%
Zero-Day Vulnerability Detection	Low	Medium	High	High
Complex Dependency Analysis	High	Moderate	High	Very High
Ease of Integration	High	Medium	High	Low
Scalability	High	Medium	High	Very High

Based on the comparative analysis of SAST, DAST, IAST, and BAML across several key metrics, BAML emerges as the most effective, particularly in environments requiring advanced threat detection, minimal false positives, and rapid response.

It excels in true positive rates (94%), low false positives (11%), and provides the highest code coverage (93%), making it ideal for complex architectures due to its superior capability in analyzing complex dependencies. IAST also performs well, especially in zero-day vulnerability

detection and code coverage, but its integration challenges may limit its applicability. While SAST and DAST are easier to integrate and offer decent scalability, their lower effectiveness in zero-day threat detection and higher false positives make them less suitable for dynamic or complex environments compared to BAML and IAST.

The development of Behavioral Analysis with Machine Learning (BAML) has demonstrated significant promise in enhancing cybersecurity. Future research could focus on several key areas to further improve BAML:

- Enhanced Data Collection: Integrating more diverse data sources, such as IoT devices and mobile applications, to enrich the training datasets;
- Real-time Adaptation: Developing advanced algorithms for real-time threat adaptation, possibly utilizing reinforcement learning;
- Scalability and Performance: Exploring distributed computing and advanced data processing to manage larger datasets more efficiently;
- Integration with Security Frameworks: Ensuring seamless integration with existing tools like SIEM systems and intrusion detection systems;
- User Behavior Analysis: Expanding analysis to include detailed user behavior for better insider threat detection;
- Compliance and Regulatory Alignment: Assisting organizations in meeting standards such as GDPR, HIPAA, and PCI-DSS through automated reporting and audit trails;

Addressing these areas will enhance BAML's capabilities, ensuring it remains at the forefront of cybersecurity innovation.

6. Conclusions.

Therefore, we conclude that our research presented in this paper demonstrates the outstanding progress made in the field of cybersecurity through the use of Behavioral Analysis with Machine Learning (BAML). Extensive experimental validation of BAML has demonstrated its ability to significantly improve the detection and prevention of software vulnerabilities. This is an improvement over traditional techniques like Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST), and Interactive Application Security Testing (IAST).

The experimental studies conducted highlight BAML's superior performance in several key areas:

- Detection Accuracy: BAML achieved the highest true positive rates, effectively identifying 94% of actual vulnerabilities, which is an improvement over other tested methods.
- Reduction of False Positives: BAML recorded the lowest false positive rates at 11%, demonstrating its precision in distinguishing between genuine threats and non-threats.
- Comprehensive Coverage: With a code coverage of 93%, BAML proved its effectiveness in analyzing a wide array of software structures and complexities.

- Speed of Detection: The ability to perform real-time analysis allows BAML to detect vulnerabilities as they occur, providing a crucial advantage in fast-paced development environments.

These results validate the hypothesis that integrating machine learning with behavioral analysis significantly enhances the capacity to identify both known and emerging vulnerabilities. The dynamic nature of BAML allows it not only to adapt to new threats but also to anticipate potential vulnerabilities through continuous learning and adaptation to changing software behaviors.

Furthermore, BAML's approach aligns with current trends in software development practices, such as Agile and DevOps, by supporting continuous integration and deployment pipelines. This alignment ensures that security testing keeps pace with rapid development cycles, embedding essential security checks within every phase of software development and deployment.

In conclusion, Behavioral Analysis with Machine Learning stands out as a potent tool in the arsenal of cybersecurity defenses, offering enhanced predictive capabilities and operational efficiency. As cyber threats evolve in complexity and subtlety, adopting advanced techniques like BAML is crucial for developing resilient digital systems capable of defending against and adapting to the cybersecurity challenges of tomorrow.

References

[1] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, K. Bobrovnikova, A Technique for the Botnet Detection Based on DNS-Traffic Analysis, Communications in Computer and Information Science, 2015. 522, 127-138.

[2] S. Lysenko, O. Pomorova, O. Savenko, A. Kryshchuk and K. Bobrovnikova, DNS-based Anti-evasion Technique for Botnets Detection, Proceedings of the 8-th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw (Poland), September 24–26, 2015. – Warsaw, 2015. – Pp. 453–458.

[3] M. C. Sánchez, J. M. C. de Gea, J. L. Fernández-Alemán, J. Garceran, A. Toval, "Software vulnerabilities overview: A descriptive study," in Tsinghua Science and Technology, vol. 25, no. 2, pp. 270-280, April 2020, doi: 10.26599/TST.2019.9010003.

[4] R. Amankwah, P. Kudjo, S. Yeboah, Evaluation of Software Vulnerability Detection Methods and Tools: A Review. International Journal of Computer Applications. 2017, 169. 22-27. doi:10.5120/ijca2017914750.

[5] A. Anwar, A. Khormali, J. Choi, H. Alasmary, S. J. Choi, S. Salem, D. Nyang, D. Mohaisen, Measuring the Cost of Software Vulnerabilities, SESA EAI, 2020, DOI: 10.4108/eai.13-7-2018.164551.

[6] What is Static Application Security Testing (SAST) (2024). URL: https://www.opentext.com/what-is/sast

[7] L. Jinfeng, Vulnerabilities Mapping based on OWASP-SANS: a Survey for Static Application Security Testing (SAST). Annals of Emerging Technologies in Computing, 2020, doi: 10.48550/arXiv.2004.03216.

[8] What is Dynamic Application Security Testing (DAST) (2024). URL: https://www.opentext.com/what-is/dast

[9] T. Sultan, S. Hendaoui. Advancing Network Security: Enhancing Dynamic Vulnerability Detection in Secure and Insecure Programming through SDN-ML Hybrid Architecture, September 08, 2023, doi: 10.21203/rs.3.rs-3318480/v1.
[10] Interactive Application Security Testing (IAST) (2023). URL: https://www.synopsys.com/glossary/what-is-iast.html
[11] S. Pargaonkar, Advancements in Security Testing: A Comprehensive Review of Methodologies and Emerging Trends in Software Quality Engineering. International Journal of Science and Research (IJSR), 2023, 12(9), 61-66.
[12] Runtime Application Self-Protection (RASP) (2023). URL: https://www.crowdstrike.com/cybersecurity-101/cloud-security/runtime-application-self-protection-rasp
[13] A. C. Eberendu, V. I. Udegbe, E. O. Ezennorom, A. C. Ibegbulam, T. I. Chinebu A Systematic Literature Review of Software Vulnerability Detection, European Journal of Computer Science and Information Technology, Vol.10, No.1, pp.23-37., 2022, doi: 10.37745/ejcsit/vol10.no1.pp23-37.
[14] Y. Valdés-Rodríguez, J. Hochstetter-Diez, J. Díaz-Arancibia, R. Cadena-Martínez, Towards the Integration of Security Practices in Agile Software Development: A Systematic Mapping Review. Appl. Sci. 2023, 13, 4578. doi: 10.3390/app13074578
[15] Z. Li, D. Zou, S. Xu, H. Jin, Y. Zhu and Z. Chen, SySeVR: A Framework for Using Deep Learning to Detect Software Vulnerabilities. IEEE Transactions on Dependable and Secure Computing, 19 4 (2022) 2244-2258
[16] A Ozment, Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models, in: Gollmann D, Massacci F, Yautsiukhin A (Eds.), Advances in Information Security: Security Measurements and Metrics, Springer, New York, NY, USA, 2006, pp. 25-369780387365848.
[17] S. Kasturi, X. Li, J. Pickard, P. Li. Prioritization of Application Security Vulnerability Remediation Using Metrics, Correlation Analysis, and Threat Model. Am J Softw Eng Appl. 2024;12(1):5-13. doi: 10.11648/j.ajsea.20241201.12.
[18] G. Markowsky, O. Savenko, S. Lysenko, A. Nicheporuk, The technique for metamorphic viruses' detection based on its obfuscation features analysis, CEUR-WS, 2018. 2104. 680–687.
[19] S. M. Ghaffarian, H. R. Shahriari, Software Vulnerability Analysis and Discovery Using Machine-Learning and Data-Mining Techniques: A Survey. ACM Comput. Surv. 50, 4, Article 56, July 2018, 36 pages. doi: 10.1145/3092566.
[20] V. Lenarduzzi, F. Pecorelli, N. Saarimaki, S. Lujan, F. Palomba, A critical comparison on six static analysis tools: Detection, agreement, and precision, Journal of Systems and Software, Volume 198, 2023, 111575, ISSN 0164-1212, doi: 10.1016/j.jss.2022.111575.
[21] F. M. Tudela, J.-R. B. Higuera, J. B. Higuera, J.-A. S. Montalvo, and M. I. Argyros, On combining static, dynamic and interactive analysis security testing tools to improve OWASP top ten security vulnerability detection in web applications, Appl. Sci., vol. 10, no. 24, p. 9119, Dec. 2020, doi: 10.3390/app10249119.
[22] A. L. S. Pupo, J. Nicolay, E. G. Boix, Deriving static security testing from runtime security protection for web applications, The Art, Science, and Engineering of Programming 6 (1), July 2021, doi: 10.22152/programming-journal.org/2022/6/1.
[23] G. Koala, D. Bassol´e, T. Tiendrebeogo and O. Si´e, Study of an ApproachBased on the Analysis of Computer Program Execution Traces for theDetection of Vulnerabilities. In: Mambo, A.D., Gueye, A., Bassioni, G.(eds) Innovations and Interdisciplinary Solutions for Underserved Areas.InterSol 2022, doi: 10.1007/978-3-031-23116-2_8.

[24] A. Takanen, P. Vuorijärvi, M. Laakso, et al. Agents of responsibility in software vulnerability processes. Ethics and Information Technology 6, 93–110 (2004), doi: 10.1007/s10676-004-1266-3.
[25] S. Zhang, D. Caragea, X. Ou, An empirical study on using the national vulnerability database to predict software vulnerabilities. In Proceedings of the International Conference on Database and Expert Systems Applications, Linz, Austria, 31 August–4 September 2011; pp. 217–231., doi: 10.1007/978-3-642-23088-2_15.
[26] R. Croft, D. Newlands, Z. Chen, and M. A. Babar, An empirical study of rule-based and learning-based approaches for static application security testing, in Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2021, pp. 1–12, doi: 10.1145/3475716.3475781.
[27] D. De silva, P. Samarasekara, R. Hettiarachchi, A Comparative Analysis of Static and Dynamic Code Analysis Techniques. TechRxiv. Preprint, 2023, doi: 10.36227/techrxiv.22810664.v1.
[28] F. Andersson and A. Öberg, Predicting Vulnerabilities in Third Party Open-Source Software using Data Mining and Machine Learning Techniques, 2023. doi: urn:nbn:se:liu:diva-195811
[29] J. D. Pereira, N. Ivaki and M. Vieira, Characterizing Buffer Overflow Vulnerabilities in Large C/C++ Projects, in IEEE Access, vol. 9, pp. 142879-142892, 2021, doi: 10.1109/ACCESS.2021.3120349.
[30] B. Wan, C. Xu, J. Koo, Exploring the Effectiveness of Web Crawlers in Detecting Security Vulnerabilities in Computer Software Applications. International Journal of Informatics and Information Systems, 2023, 6(2), 56-65. doi:10.47738/ijiis.v6i2.158
[31] O. Pomorova, O. Savenko, S. Lysenko, A. Kryshchuk, A.Nicheporuk A Technique for detection of bots which are using polymorphic code, Communications in Computer and Information Science. 2014. 431, 265-276
[32] O. Pomorova, O. Savenko, S. Lysenko, A. Nicheporuk, Metamorphic Viruses Detection Technique based on the the Modified Emulators, CEUR Workshop Proceedings, 1614 (2016) 375-383.
[33] A. Kashtalian, S. Lysenko, O. Savenko, A. Nicheporuk, T. Sochor , V. Avsiyevych, (2024). Multi-co mputer malware detection systems with metamorphic functionality. Radioelectronic and Computer Systems, 2024(1), 152-175. doi: 10.32620/reks.2024.1.13
[34] O. Savenko, A. Nicheporuk, S. Lysenko, I. Hurman, Dynamic signature-based malware detection technique based on API call tracing CEUR Workshop Proceedings, 2393 (2019) 633-643.
[35] E. M. Cherrat, R. Alaoui, H. Bouzahir, Score fusion of finger vein and face for human recognition based on convolutional neural network model. *International Journal of Computing*, *19*(1), 2020, 11-19. https://doi.org/10.47839/ijc.19.1.1688