Literature Review on Real-Time Threat Detection and Malware Classification

Faheema Hassan Kudukky[1], Swathi Ravi M[2], Lajin C P[3], Dr. Reema Mathew A[4]
[1,2,3] *Department of CSE Cyber Security, Vimal Jyothi Engineering College Chemperi, Kannur*
[4]*Professor, Department of CSE Cyber Security, Vimal Jyothi Engineering College Chemperi, Kannur*

***Abstract*—The extensive use of open-source software and third- party components has greatly increased the attack surface of modern applications. Organizations are exposed to not only sup- ply chain attacks (in the form of backdoors), but also to hidden malware and unpatched vulnerabilities. High-profile incidents, including the use of backdoors found in some of the most popular utilities in the world, have demonstrated how serious these risks can be and how challenging it is to maintain software integrity. Meanwhile, traditional detection methods are still being under- mined by an ever-growing number of sophisticated malware variants. As a result, a timely response and classification are very much needed. Thus, real-time monitoring is the key to the exchange of security risk, anomalies and crashes before they lead to extensive disruptions. The detection of threats and malware classification is requisite in improving system resiliency, reducing the impact on enterprises and strengthening their proactive defense against cyber threats.**

***Index Terms*—Real-time monitoring, Malware Classification, Threat Detection, Supply Chain Attacks, Proactive Defense, Cyber Threats.**

I. INTRODUCTION

The current world scenario shows that digital systems are becoming increasingly vulnerable to cyber breaches, data privacy violations and disruptions in critical infrastructure. With the growing interdependencies between technologies and the rising sophistication of cyberattacks, conventional security solutions are quickly becoming reactive in nature and failing to keep pace with changing threats. The surge in high-profile data breaches, ransomware incidents and supply chain compromises has revealed the fragility of traditional defenses, which are predominantly software-based and limited in their scope. Attackers now exploit advanced techniques such as AI-driven threats, zero-day exploits and malicious code injections, thereby intensifying the urgency for more proactive and adaptive security systems.

In view of these evolving threats and the increasing reliance on digital systems across industries, the demand for robust, scalable and cost-effective solutions has become critical. Conventional one-dimensional defense mechanisms are no longer sufficient, as they primarily focus on detection after an incident has occurred rather than preventing it in real time. To secure sensitive data and safeguard digital infrastructures, future systems must adopt a multi-layered defense strategy capable of continuous monitoring, real-time risk detection and automated mitigation. Only through such proactive and adaptable approaches can organizations reduce attack surfaces, anticipate emerging risks and ensure resilience against com- plex, persistent and continually evolving cyber threats.

II. LITERATURE SURVEY

Joseph Merigala et al. [1] provide a thorough analysis of new supply chain attacks, using the XZ Utils backdoor (CVE-2024-3094) as a prime example. The paper outlines how a threat actor, who went by the alias Jia Tan, built trust in the open-source community over a number of years, obtained maintainer access and delivered a backdoor that had an effect on a number of commonly used linux components. The attack approach exploited subtlety by injecting malicious payloads into test files and build configuration, allowing the indicators of the backdoor to avoid normal code review and antivirus inspection. The study further delves into social factors, such as maintainer fatigue and lack of project oversight, as important facilitators of the attack. The authors further discuss the potential reach and impact of the back door - that is, remote access to innumerable Linux systems - and insist on a need for increased

software provenance verification, increased vetting of contributors, continuous monitoring and cross-enterprise threat intelligence sharing. Apart from this specific case, the

TABLE I COMPARISON TABLE

Ref	Title	Methodology	Advantages	Disadvantages
[1]	Analysis of Supply Chain Attacks in Open Source Software and Mitigation Strategies	Tracing backdoor changes and maintainer manipulation, CVSS-based analysis.	- Real world case study with social and technical insights. - Actionable recommendations for OSS security.	- Single case focus; may not generalize. - Solutions are mostly conceptual.
[2]	An analysis of malicious behaviors of open source pack- ages using dynamic analysis.	Sandbox execution with run-time behavior logging.	- Captures runtime actions missed by static tools.	- Output is raw and needs expert review
[3]	A Multi-Dimensional Visual Analytics Tool for the Security Posture of Open-Source Soft- ware	Visual dashboard; integrates NVD + GitHub commits with SZZ algorithm.	- Multi-dimensional view of risks. - Developer validation shows im- proved understanding.	- Relies heavily on data availability. - Only assessment, no remediation.
[4]	VulnEx: Exploring OpenSource Software Vulnerabilities in Large Development Organizations to Understand Risk Exposure	Visualization tool (VULNEX) using Eclipse Steady scans (static/dynamic).	- Organization-wide view of OSS vulnerabilities. - Easy visual tables for risk analysis.	- Doesn't scale well for very large orgs. - Limited to Java/Python (Eclipse Steady).
[5]	Malicious Source Code Detection Using Transformer	Transformer based model using static code analysis, embeddings, and anomaly detection.	- High precision (0.909) and scalability.	- Static-only; misses runtime threats. - Needs large labeled datasets.
[6]	Supporting the Detection of Software Supply Chain Attacks through Unsupervised Signature Generation	AST + Markov Clustering; unsupervised signature generation.	- Very high accuracy (F1 0.99). - Reduced manual review burden.	- May miss entirely new attack types. - Requires frequent updates.
[7]	Fixing Security Vulnerabilities with AI in OSS-Fuzz	LLM agent (CodeRover-S) integrated with fuzzing crash re- ports.	- Couples detection and remediation. - Emphasizes exploit validation.	- Technology still immature. - Real world deployment challenges.
[8]	Enhanced Malware Image Classification through Ensemble Model	Malware binaries to images; ensemble of EfficientNetB0 + EffiCBNet.	- Higher accuracy than single models. - Reduces bias, overfitting.	- High computational demands. - Not suitable for low-resource.
[9]	Image-based malware representation approach with EfficientNet convolutional neural networks for effective malware classification	EfficientNetB1 CNN with fixed-width malware images for accuracy and efficiency.	- Good trade-off between accuracy and efficiency. - Captures subtle structural details.	- Needs large, curated datasets. - Vulnerable to adversarial evasion.
[10]	Binary Malware Image Classification Using Machine Learning with Local Binary Pattern	Converts binaries to images, extracts LBP texture features, classifies with TensorFlow ML.	- Resilient to code obfuscation. - Enables fast automated analysis.	- Can confuse overlapping malware families. - Misses behavioral information.

[11]	Co-training for Image-Based Malware Classification	Semi-supervised co-training with grayscale malware images, LBP features and dual classifiers.	- Cuts annotation costs. - Noise-learning improves accuracy.	- Sensitive to poor unlabeled data. - More complex training pipeline.
[12]	Image-Based Malware Detection Using α-Cuts and Binary Visualisation	Applies α-cuts for segmentation, creates sparse matrices, classifies with ResNet50 CNN.	- Enhances interpretability and speed. - Supports privacy by masking data.	- Generalization to diverse sets is hard. - Needs careful preprocessing.
[13]	Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning.	Converts binaries to images, applies deep CNN with transfer learning for family classification.	- High F1 score (0.99). - Shows benefits of custom design.	- Transfer learning less effective for malware. - Requires more design effort.
[14]	Hybrid Malware Classification Method Using Segmentation-Based Fractal Texture Analysis and Deep Convolution Neural Network Features	Combines texture features (SFTA) with CNN features (AlexNet/Inception-V3) using SVM, KNN, DT.	- Fusion improves accuracy and balance. - More robust to variant changes.	- Increased complexity and integration effort. - Risk of overfitting.
[15]	A Malware Detection Method of Code Texture Visualization Based on an Improved Faster RCNN Combining Transfer Learning	Uses code-texture images with Faster R-CNN + transfer learning for feature localization.	- Strong zero-day, obfuscation results. - Visualization with deep learning.	- Needs large labeled data and compute. - Vulnerable to adversarial evasion.

paper reviews broader directions for mitigating actions of this nature and emphasizes that ensuring aggressive security for open source requires not only technical measures but also community governance and vigilance on the part of users.

Thanh-Cong Nguyen et al. [2] propose a comprehensive dynamic analysis framework for detecting malicious behaviors in open-source software packages, a problem that has become increasingly urgent with the rise of software supply chain attacks such as the recent backdoor incident in the widely used xz utility. Unlike conventional vulnerability scanners that rely heavily on static analysis and focus primarily on identifying already-documented weaknesses, their approach emphasizes observing how packages behave during execution, allowing researchers to uncover runtime activities that often remain hidden at the code level. The study spans multiple ecosystems, including npm, PyPI, RubyGems and crates.io and reveals that malicious packages consistently exhibit distinct patterns compared to benign ones: they are far more likely to initiate outbound communications with suspicious domains, execute arbitrary system-level commands and employ obfuscation techniques to disguise their true purpose. Interestingly, these techniques are often unsophisticated—attackers frequently use simple methods such as base64 encoding or rely on standard tools like curl and wget for data theft—yet they remain effective in the absence of robust monitoring. The authors argue that dynamic analysis provides richer insights than static approaches by capturing real execution traces, but they also acknowledge that existing tools are still underdeveloped, typically producing raw and unstructured outputs that demand expert interpretation before they can be operationalized. To address this limitation, they suggest that future work should focus on automated rule generation for suspicious behaviors and on integrating these tools into broader supply chain security processes so that registries, developers and organizations can proactively defend against emerging threats.

Tianyu Li et al. [3] recognize that open-source projects are integral to modern development but often lack clear indicators of security health. Their paper introduces a visual analytics dashboard designed to provide a comprehensive assessment of project security posture. Unlike traditional tools that focus

only on code vulnerabilities, this dashboard integrates multiple data dimensions: commit histories, contributor behavior and vulnerability timelines. It uses data from GitHub and the National Vulnerability Database (NVD), combined with the SZZ algorithm to identify vulnerability- inducing and fixing commits. Two case studies—Odoo and ImageMagick—demonstrate the dashboard's ability to handle large datasets (180,090 and 22,484 commits respectively) and track hundreds of vulnerabilities. Visualizations help users identify risky periods of development, inactive or overly active contributors and long-lived vulnerabilities. An evaluation with experienced developers showed that the tool significantly improves the ability to interpret project risks quickly, making it useful for maintainers, auditors and even non-technical stakeholders. The authors conclude that multi-dimensional, visual approaches are essential for understanding both technical and social factors that influence open-source security.

Frederik L. Dennig et al. [4] present VulnEx, a visual analytics tool designed to explore and assess open-source software (OSS) vulnerabilities at an organization-wide scale. The study emphasizes that while OSS accelerates development, it also introduces security risks through widely reused vulnerable dependencies. Traditional static and dynamic code analysis tools often operate at the level of individual applications and fall short in giving organizations a holistic view of risk exposure across large codebases. To address this gap, VulnEx introduces three complementary table-based representations—repository, library and vulnerability views—that enable security analysts to trace vulnerabilities across software projects, prioritize risks based on severity (e.g., CVSS scores) and identify critical third-party libraries requiring updates. The system supports drill-down analysis through a hierarchical dependency tree, a CVE matrix and meta-information such as GitHub activity and quality metrics. Applied to hundreds of Eclipse Foundation repositories, VulnEx demonstrated its ability to uncover widely spread critical vulnerabilities like those in activemq-all and tomcat-embed-core, which affected multiple repositories. Feedback from SAP security experts validated its usefulness for visualizing risk exposure, while also pointing out future scalability and contextual information needs. By bridging auditing requirements with interactive visualization, the paper highlights VulnEx as a step toward more effective vulnerability management in large-scale OSS-based organizations.

Chen Tsfaty et al. [5] address software supply chain risks by detecting malicious injections in open-source codebases. Their paper highlights that supply chain attacks are rising because attackers exploit the trust in widely used packages. They introduce MSDT, a static analysis framework that leverages modern natural language processing techniques—specifically transformer models like Code2Seq—to analyze source code. MSDT works by embedding each function in a semantic vector space, then applying clustering algorithms (DBSCAN) to detect anomalies that may indicate injected malicious code. This research proposes utilizing a large dataset containing over 600,000 functions and providing real malicious payloads for the purpose of testing performance. MSDT achieved precision@k scores of up to 0.909 showing that the algorithm can accurately detect suspicious code fragments in a large dataset. The authors also benchmarked MSDT against commonly used static analysis tools and found that focusing pointwise at the function level enables finer-grained detection while reducing false-positives. This paper also highlighted the need to adapt deep learning models to program analysis and found that transformers are able to generalize between programming languages, making them suitable for multilanguage security scanning. Overall, this research provides evidence that AI-driven static analysis can improve defenses against contemporary methods employed in software supply chain attacks.

Marc Ohm et al. [6] emphasize general software supply chain attacks, specifically the issue of trojanized packages that are hiding in repositories such as npm, or PyPI. The authors observed that trojanized packages go unnoticed for months, allowing the threat actors to fully compromise downstream applications. The authors introduce a tool called ACME, which is able to automate signature generation, helping to reduce the burden of manual code review. Using ASTs to analyze packages, ACME was able to create clusters of related malicious packages. The authors utilized the Markov Cluster Algorithm (MCL) to mimic classifications made by experts. From these clusters, it derives characteristic code fragments that become signatures for future scanning. The paper evaluates ACME against a curated dataset and achieves an F1 score of 0.99, showing its ability to

replicate expert decisions with minimal human effort. Importantly, the system discovered six previously unreported malicious packages, which were later removed from npm. The authors stress that automation is vital because attackers frequently reuse or slightly modify code fragments to evade detection. The paper demonstrates that unsupervised techniques can scale to large repositories and give security teams actionable insights much earlier in the attack lifecycle.

Zhang et al. [7] focus on the critical issue of vulnerability remediation in open-source software systems, situating their study within the context of OSS-Fuzz, one of the most significant large-scale fuzzing infrastructures. While fuzzing has been highly successful at detecting vulnerabilities, including the discovery of more than 10,000 flaws across a wide range of projects, the authors emphasize a persistent problem: many identified vulnerabilities remain unpatched due to the manual burden of remediation. Their study investigates whether recent advances in AI-driven programming agents, particularly those inspired by large language models, can be adapted to address this gap. They argue that autonomous patch generation has the potential to complete the software protection cycle by coupling detection with timely remediation. Importantly, their findings stress that evaluating patches on surface similarity to developer fixes is insufficient, since real security assurance requires validation against exploit inputs and execution contexts. By shifting focus from code similarity metrics to dynamic validation, the study underscores the need for more rigorous standards of automated vulnerability fixing. The authors reveal further implications - for the future of development, effectively automating remediation could greatly reduce developer workload, reduce exposure time and enhance ecosystem resilience as a whole. But they also mention there are still challenges because automated patching is still an emerging practice that has to balance autonomy and accuracy if it is to garner trust from developers. So their contribution lies in their practical experimentation, but also in contextualizing the future of automated remediation as a part of a secure software development life cycle.

Eshwar Kotha et al. [8] propose an ensemble-based approach for malware image classification that combines the predictive strengths of multiple deep learning models into a unified framework, thereby achieving consistently higher accuracy than individual classifiers and demonstrating particular effectiveness in distinguishing between diverse malware families, with the study underscoring that such ensemble techniques not only reduce the weaknesses and biases inherent in single-model architectures but also contribute to greater overall robustness and resilience against variations in malware representation, while at the same time acknowledging that this improvement comes at the cost of increased computational demands and more complex training procedures, which may hinder scalability in resource- constrained environments, leading the authors to emphasize that future research should concentrate on designing optimized ensemble strategies that balance accuracy with efficiency, explore methods to streamline computation without sacrificing detection performance and ultimately extend the applicability of their framework to real-time malware detection scenarios where both speed and reliability are critical for practical deployment.

Rajasekhar Chaganti et al. [9] propose an image-based malware representation framework that leverages the scalable architecture of EfficientNet convolutional neural networks for classification, arguing that Efficient Net's compound scaling method, which simultaneously balances network depth, width and resolution, enables superior feature extraction from malware images compared to traditional CNN approaches, thereby allowing the model to capture subtle structural patterns and visual signatures that are often overlooked by less sophisticated architectures and through extensive experimentation they demonstrate that this approach achieves high detection accuracy while consuming fewer computational resources, making it not only effective in identifying diverse malware families but also efficient enough to be considered for deployment in operational environments where resource constraints are a major concern, yet despite these promising outcomes, the authors carefully acknowledge certain limitations inherent to their method, most notably the dependency on large-scale, high-quality datasets for training, which raises challenges regarding data availability and curation, as well as the model's sensitivity to adversarially crafted inputs that can exploit weaknesses in learned representations and undermine detection performance and in light of these challenges they propose that future research directions should prioritize the design of adversarially robust variants of EfficientNet, the exploration of advanced

regularization or defense mechanisms to resist evasion attacks and the construction of broader cross-platform and heterogeneous datasets that incorporate samples from multiple operating systems and malware categories so as to enhance the generalizability, resilience and real-world applicability of deep learning–based malware detection systems.

Luo and Lo [10] presents a new approach to malware classification that represents the emerging trend of utilizing visualization in cybersecurity. The authors contend that, although static and dynamic analyses offer value for classifying malware, they face challenges such as susceptibility to obfuscation and behavior based on the parameters of execution environment. As a remedy, they explore the representation of malware binaries as grayscale images that can be processed with texture-based descriptors. From this perspective, the authors can leverage visual and structural features present in a family of malware, making classification less reliant on an analysis of explicit code. The authors demonstrate how texture-based recognition, especially using local binary patterns as descriptors, can uncover subtle differences between malware variants that may otherwise exhibit similar behaviors or binary codes. Although the authors document increased classification performance in their approach over traditional descriptor-based analyses, they acknowledge several challenges remain. These include creating classifiers that improve their performance across malware families that share structural similarities and the ongoing evolution of malware that generates new visual features at a fast pace. Still, the authors reinforced the point that image-based methods represent a useful additional classification strategy and ultimately, their findings show value in utilizing visualization with machine learning to further the classification toolset in malware detection, especially in situations where fast and automated classification is needed.

Tan Gao et al. [11] propose a semi-supervised co-training framework for image-based malware classification that seeks to address one of the most persistent challenges in the field—the heavy dependency on large, meticulously labeled datasets that are costly and time-consuming to construct—by introducing a pipeline in which malware binaries are first transformed into grayscale images so that their structural and behavioral information can be visually encoded, then subjected to texture feature extraction that captures subtle spatial distributions and repetitive patterns characteristic of different malware families and finally processed through a collaborative learning mechanism that combines insights from both labeled and unlabeled data to build a more complete decision boundary, with the novel integration of noise-learning theory serving as a safeguard against the accumulation of errors typically introduced by pseudo-labeling strategies, thereby allowing the framework to iteratively refine itself and progressively enhance its detection performance over time, as demonstrated in a series of controlled experiments where the proposed method not only achieved superior classification accuracy compared to conventional supervised models but also demonstrated substantial reductions in annotation costs, underscoring its potential practicality in real-world environments where labeled samples are scarce, yet despite these promising outcomes the authors acknowledge that the framework's success remains contingent on the quality and representativeness of the extracted features as well as the careful management of unlabeled data that, if handled incorrectly, could propagate noise and degrade performance, leading them to recommend that subsequent research should concentrate on refining advanced noise-filtering strategies, exploring adversarially robust learning mechanisms that can withstand attempts at data poisoning or evasion, incorporating diverse and heterogeneous datasets drawn from multiple operating systems and malware families to enhance the model's generalizability and ultimately extending the framework into large-scale, real-world deployment scenarios where the balance between accuracy, efficiency, scalability and resilience against emerging threats is crucial for the advancement of malware detection technologies.

Saridou et al. [12] provide an introduction of binary visualization as an extension of malware detection within cybersecurity research. They contextualize their research in relation to the transition from signature-based detection, which is ineffective against obfuscated or fast changing malware, towards visually and learning-based classification systems. The authors discuss how converting binaries to images creates distinguishable structural and texture-based nuances that would not be recognized from traditional analysis of the binary. Their innovative aspect is the focus on α-cuts from fuzzy set theory that transform pixel values to sparse representations to better segment,

as well as lessen computation times. This means improving the visualizations of malware as well as wider operational costs - both important considerations for real-time intrusion detection systems analyzing a large volume of data. However, they argue that the graphic method is also aligned with privacy-preserving methods because inherently, visualization of sensitive data can remain obfuscated and still form useful features for classification. The authors evidence that systems employing visualization and better segmentation are advantageous of performing either equal to or better than conventional baselines, thus they should be considered a consideration for supporting operational security workflows. The authors acknowledge that the limitations of their work acknowledges that models should be expected to generalize across a diversifying agent.

Dipendra Pant et al. [13] examine the classification of malware by converting malware binaries into grayscale images and using detection methods based on computer vision. This article explains that malware samples often have unique structural and behavior patterns and it is possible to capture these features visually, therefore easing classification. This paper investigates four deep learning models: pretrained VGG16, ResNet-18, InceptionV3 and the authors custom CNN. Although transfer learning (VGG16) achieved 88.40% accuracy, the custom CNN is the most superior network, which achieved 98.7% validation accuracy and 0.99 F1 score. The model architecture and parameter values were optimized; thus, the authors show an image representation of a geometric computer program can capture spatial specific features that are lost with traditional static malware analysis. The paper also recognizes that transfer learning will have limited success when the domain of the source images are very different to the target domain (binary malware data). Furthermore, the paper recognizes by comparing the different architectures, custom models and optimizing for the domain will be crucial in achieving a higher accuracy. Finally, the authors concluded that image based malware analysis is an interesting avenue of malware research and if rapid inference engines are used with rapid filtered classification, it is valuable for real time detection.

Nisa et al. [14] present a hybrid malware classification framework that combines deep learning with traditional feature analysis for better accuracy and robustness in malware detection. Their work recognises two significant obstacles to malware research: the need to extract stable and informative features and the problem of dataset imbalance adversely affecting classification. They combine deep learning via convolutional neural networks that can learn complicated features based on experience from image based data, with segmentation based fractal texture analysis learned features that take into account structure that a network could not learn on its own. This multimodal fusion approach allows the frameworks to retain more informative features which leads to improved performance across many families of malware. The authors frame their work in the context of computer science's avant-garde trend of leveraging visual outputs for malware detection and they note the reduction of binary files through the use of grayscale images that preserve data structure while permitting better feature extraction. They validate the approach and demonstrate high accuracy with manually checking multiple malware instances and documentation of the effort with regards to demonstrating robustness against class imbalance particularly where augmentation was needed and enhanced performance in predictably incorrect classifications. While the authors noted that malware would always evolve, even with high accuracy the models in use must accept some tolerance in terms of new variants and forms of attack. Their research ultimately emphasises the value in hybrid approaches that fuse analytical level and go further in hybrid perspectives of features.

Yuntao Zhao et al. [15] propose a novel malware detection method that transforms binary code into texture images and applies an improved Faster R-CNN model with transfer learn- ing for classification, where the transformation of binaries into texture images enables the model to capture rich structural and spatial features of malware that are often overlooked by traditional static and dynamic analysis methods, thereby addressing the limitations of conventional approaches that typically rely on handcrafted features or runtime behavior which may be incomplete or easily obfuscated by attackers and through this visualization-driven deep learning framework the authors report achieving not only high detection accuracy but also notable efficiency in handling complex malware families, with particular strength in detecting zero-day malware samples that lack prior signatures, demonstrating the capability of

the model to generalize across unseen threats; furthermore, the study emphasizes that visualization-based techniques, when effectively combined with advanced deep learning architectures like Faster R-CNN, have the potential to significantly enhance malware detection performance compared to baseline machine learning classifiers, yet it also carefully acknowledges several persisting challenges including the necessity of large, well-labeled datasets to fully train and validate the models, the considerable computational costs associated with deploying deep neural networks in real-time cybersecurity environments and the growing risk of evasion through adversarial attacks that deliberately manipulate input representations to mislead detection systems, thus motivating future research directions that should prioritize the design of lightweight and resource- efficient models suitable for deployment in practical systems, the integration of robust adversarial defense mechanisms to counter potential attacks and the expansion of broader and more diverse datasets that can better represent real-world mal- ware distribution, thereby ensuring the scalability, resilience and overall robustness of visualization-driven malware detection frameworks in increasingly complex and adversarial cyber environments.

CONCLUSION

This review of literature portrays a clear technological convergence towards intelligent, resilient and adaptive cybersecurity, driven by the parallel evolution of open-source software supply chain security, deep learning–based malware detection and AI-driven automation for vulnerability analy- sis. These emergent solutions are dedicated to transcending the limitations of traditional security architectures: supply chain monitoring and visualization frameworks provide essential governance and transparency across complex dependency ecosystems where malicious packages and vulnerabilities are increasingly injected; transformer-based and AI-augmented fuzzing tools accelerate the discovery and remediation of security flaws, offering enterprises proactive means to secure their open-source adoption; and image- or texture-based malware classification through CNNs, EfficientNet, transfer learning, hybrid fractal-texture analysis and ensemble learning solves the critical bottleneck of detecting polymorphic and evasive malware that routinely bypasses legacy signature-based systems. Although much promising potential abounds, significant challenges remain. Supply chain security mechanisms continue to struggle with scalability, fragmented adoption across organizations and the difficulty of achieving standardization in heterogeneous open-source communities. AI-driven fuzzing and malicious code detection, while powerful, must contend with issues of interpretability, false positives and the risks of overfitting to known attack patterns rather than generalizing to novel threats. Likewise, advanced malware classification pipelines, though superior in accuracy, face practical hurdles such as adversarial evasion, limited robustness across diverse malware families, high training costs and the need for explainability in security-critical decision making. As cyber threats become more dynamic, adaptive and distributed, the integration of these paradigms—open-source supply chain defense, AI-augmented vulnerability discovery and intelligent malware classification—represents a promising direction for building security systems that are both proactive and adaptive, capable of evolving alongside adversaries. Yet their effective- ness ultimately depends on overcoming persistent obstacles in scalability, interpretability and operational integration, while also ensuring transparency and trust through human-in-the-loop oversight. The unifying vision emerging from these works is the gradual movement towards a cybersecurity ecosystem where automation, artificial intelligence and collaborative defense mechanisms work in synergy to provide resilient, explainable and scalable protection for increasingly complex digital infrastructures.

REFERENCE

[1] J. Merigala, V. Kumar, J. Gujjarlapudi, M. Gupta, and A. S. Kumar, “Analysis of supply chain attacks in open-source software and mitigation strategies,” in *2024 5th International Conference on Communication, Computing & Industry 6.0 (C2I6)*. IEEE, 2024.

[2] T.-C. Nguyen, D.-L. Vu, and N. C. Debnath, “An analysis of malicious behaviors of open-source packages using dynamic analysis,” *Preprint*, 2024. [Online]. Available: https://www.researchgate.net/publication/382447540

[3] T. Li, C. Lu, and B. Lagaisse, “A multi-

dimensional visual analytics tool for the security posture of open-source software," in *2025 IEEE/ACM 3rd International Workshop on Software Vulnerability Management (SVM)*. Leuven, Belgium: IEEE/ACM, 2025.

[4] F. L. Dennig, D. Mendez, I. Tsalicoglou, C. Berger, and S. Rinderle- Ma, "Vulnex: Exploring open-source software vulnerabilities in large development organizations to understand risk exposure," in *2022 IEEE Symposium on Visualization for Cyber Security (VizSec)*. IEEE, 2022, pp. 1–11.

[5] C. Tsfaty and M. Fire, "Malicious source code detection using transformer," *arXiv preprint arXiv:2209.07957*, 2022. [Online]. Available: https://arxiv.org/abs/2209.07957

[6] M. Ohm, L. Kempf, F. Boes, and M. Meier, "Supporting the detection of software supply chain attacks through unsupervised signature generation," *arXiv preprint arXiv:2011.02235*, 2021. [Online]. Available: https://arxiv.org /abs/2011.02235

[7] Y. Zhang, J. Wang, D. Berzin, M. Mirchev, D. Liu, A. Arya, O. Chang, and A. Roychoudhury, "Fixing security vulnerabilities with ai in oss- fuzz," *ACM Transactions on Software Engineering and Methodology*, vol. 1, no. 1, pp. 1–21, 2024.

[8] E. Kotha, L. H. Palivela, and A. Begum, "Enhanced malware image classification through ensemble model," in *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*. IEEE, 2024, pp. 1421–1429.

[9] R. Chaganti, V. Ravi, and T. D. Pham, "Image-based malware repre- sentation approach with efficient net convolutional neural networks for effective malware classification," *Journal of Information Security and Applications*, vol. 69, p. 103306, 2022.

[10] J.-S. Luo and D. C.-T. Lo, "Binary malware image classification using machine learning with local binary pattern," in *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 4664–4669.

[11] T. Gao, X. Li, and W. Chen, "Co-training for image-based malware clas- sification," in *2021 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC)*. IEEE, 2021, pp. 568–573.

[12] B. Saridou, I. Moulas, S. Shiaeles, and B. Papadopoulos, "Image- based malware detection using α-cuts and binary visualisation," *Applied Sciences*, vol. 13, no. 7, p. 4624, 2023.

[13] D. Pant and R. Bista, "Image-based malware classification using deep convolutional neural network and transfer learning," in *Proceedings of the 2021 3rd International Conference on Advanced Information Science and System (AISS 2021)*. Sanya, China: ACM, 2021, pp. 1–6.

[14] M. Nisa, J. H. Shah, S. Kanwal, M. Raza, M. A. Khan, R. Damasˇevicˇius, and T. Blazˇauskas, "Hybrid malware classification method using segmentation-based fractal texture analysis and deep convolution neural network features," *Applied Sciences*, vol. 10, no. 14, p. 4966, 2020.

[15] Y. Zhao, W. Cui, S. Geng, B. Bo, Y. Feng, and W. Zhang, "A malware detection method of code texture visualization based on an improved faster rcnn combining transfer learning," *IEEE Access*, vol. 8, pp. 166 630–166 643, 2020.