Review

Not peer-reviewed version

AI-Powered Automated Bug Bounty Platform

Tahir Naquash , Zeeshan Yalakpalli * , Shania Margaret Saini , Shivshankar - , Ayesha Siddiqua

Posted Date: 17 June 2025

doi: 10.20944/preprints202506.1363.v1

Keywords: Vulnerability Detection; Automation; Artificial Intelligence (AI); Machine Learning (ML); Cybersecurity; Bug Bounty; Literature Survey; SQL Injection; Deep Learning; Penetration Testing

Review

AI-Powered Automated Bug Bounty Platform

Tahir Naquash [1], Zeeshan Yalakpalli [2,*], shania Margaret Saini [2], Shivshankar [2] and Ayesha Siddiqua [2]

[1] Assistant Professor, Dept. of CSE, HKBK College of Engineering, Bangalore, India; tahir.cs@hkbk.edu.in

[2] Student, Dept. of CSE, HKBK College of Engineering, Bangalore, India; 1hk22cs145@hkbk.edu.in (S.M.S.); 1hk22cs153@hkbk.edu.in (S.); 1hk22cs191@hkbk.edu.in (A.S.)

* Correspondence: 1hk22cs189@hkbk.edu.in

Abstract: Cybersecurity remains a critical challenge, demanding efficient and reliable methods for vulnerability detection. Traditional approaches often struggle with speed, scalability, and the evolving nature of threats. This survey reviews recent literature (2022-2024) focusing on automated vulnerability detection, particularly leveraging Artificial Intelligence (AI) and Machine Learning (ML). We analyze the key advancements presented in these studies, such as improved accuracy in identifying specific flaws (e.g., SQL Injection, Cross-Site Scripting) using hybrid methods and deep learning, and the increased efficiency offered by automation scripts and AI-driven penetration testing tools. However, we also critically examine the persistent limitations highlighted, including dependencies on large, high-quality datasets, the 'black-box' nature of some AI models, challenges in detecting zero-day threats, the resource-intensive nature of advanced models, and the continued need for human validation. These identified gaps and drawbacks collectively underscore the necessity for exploring more integrated, autonomous, and transparent security solutions, motivating research into systems that combine AI's detection capabilities with technologies ensuring verifiable and trustworthy reporting.

Keywords: vulnerability detection; automation; Artificial Intelligence (AI); Machine Learning (ML); cybersecurity; bug bounty; literature survey; SQL injection; deep learning; penetration testing

I. Introduction

The digital transformation across industries has transpired due to technological advancements that have led to an explosion in the number of web applications, APIs, cloud services, and IoT devices deployed by organizations. While enabling innovation and efficiency, this proliferation creates a vastly expanded attack surface, making robust and continuous security auditing more critical than ever. Traditional security paradigms are struggling to keep pace with the speed of development (DevOps) and the increasing sophistication of cyber threats.

A. Limitations of Traditional Security Auditing

Historically, security auditing relied heavily on manual penetration testing and periodic scans using conventional vulnerability assessment tools. Manual penetration testing, performed by skilled ethical hackers, offers depth and the ability to uncover complex business logic flaws but suffers from significant drawbacks. It is inherently slow, expensive, and difficult to scale across large, dynamic environments. The effectiveness heavily relies on the skill and experience of the individual tester. Traditional automated scanners, on the other hand, primarily rely on signature-based detection for known vulnerabilities. While faster, they often generate a high volume of false positives, consuming valuable time for security teams who must manually verify each alert, and struggle to identify novel (zero-day) threats, business logic flaws, or complex chained exploits. Furthermore, scaling these approaches effectively to cover the rapidly growing digital footprint of modern enterprises remains a significant hurdle.

B. Evolution Towards Intelligent Automation

The limitations of traditional methods spurred the evolution of security testing methodologies. Static Application Security Testing (SAST) tools analyze source code, bytecode, or binaries without executing the application, enabling vulnerability detection early in the software development lifecycle (SDLC). Dynamic Application Security Testing (DAST) tools interact with running applications, sending payloads and analyzing responses to identify vulnerabilities from an external perspective, similar to a black-box attacker. Interactive Application Security Testing (IAST) attempts to bridge the gap by instrumenting the application to monitor its internal state during dynamic testing, potentially offering greater accuracy and context than SAST or DAST alone. The limitations of these methods, particularly in handling complex threats and reducing false positives, have paved the way for the integration of AI/ML, as explored in this survey.

C. Focus on Key Vulnerabilities and Bug Bounties

This survey particularly considers the application of automated techniques for detecting prevalent and high-impact vulnerabilities such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and weaknesses in authentication mechanisms. These flaws continue to be exploited widely, leading to significant data breaches and system compromises. Furthermore, the rise of bug bounty programs represents a paradigm shift, leveraging crowdsourced security researchers to find vulnerabilities in exchange for rewards. While effective in harnessing diverse expertise, these programs often face challenges in managing the volume of submissions, ensuring report quality, validating findings efficiently, and providing timely feedback. The potential for AI-powered automation to enhance both the detection capabilities within bug bounty platforms and the efficiency of the management process itself is a key motivator for the research landscape surveyed here.

D. Survey Scope and Purpose

Researchers have explored AI/ML for various security functions, from threat prediction to specific vulnerability identification like SQLi and code defect analysis. Automation frameworks have also been developed to streamline processes like reconnaissance. However, despite these advancements, significant hurdles remains. The purpose of this survey is to critically evaluate recent research (focusing on 2022-2024) in automated and AI-driven vulnerability detection relevant to enhancing bug bounty effectiveness. We distill key advances and critically examine reported drawbacks and limitations. By analyzing these, we aim to highlight research gaps and technological needs motivating the exploration of novel, integrated platforms combining AI-driven auditing with mechanisms for trustworthy reporting, ultimately aiming to improve upon traditional security auditing and bug bounty models. This review covers hybrid testing, AI/ML applications, automation scripts, LLMs in penetration testing, and bug tracking, synthesizing findings to understand the field's trajectory and the impetus for future innovation.

II. Literature Review: Advances and Drawbacks

This section dissects the selected literature, focusing on the contributions and limitations relevant to automated vulnerability detection and management.

A. Hybrid and Automated Scanning Techniques

Early advancements focused on combining automated tools with manual oversight. Paper [1] exemplifies this by using OWASP ZAP alongside manual testing on the Bwapp testbed. **Advance:** Effectively covered common vulnerabilities like SQLi and XSS in a controlled environment.

Drawback: Inherently limited by the time-intensive nature of manual testing and its inability to proactively identify unknown (zero-day) threats. This highlights a need for faster, more predictive methods.

Automating the reconnaissance phase of bug hunting, as explored in Paper [6], offers significant efficiency gains. **Advance:** The Bash script integrating tools like Sublist3r, Nmap, SQLmap, and OWASP ZAP drastically reduced recon time (by 70%) and provided comprehensive coverage by combining multiple open-source tools. **Drawback:** The approach suffers from dependency on pre-installed tools and generates noise (false positives) requiring manual validation. Ethical concerns regarding aggressive scanning (e.g., potential DoS) also arise. This points towards the need for smarter automation that minimizes noise and operates within safe boundaries.

B. AI/ML in Vulnerability Detection and Cybersecurity

The integration of AI/ML represents a major thrust in modern cybersecurity research. Paper [2] provides a general overview.

Advance: AI/ML enables real-time threat detection, adaptability to evolving patterns, and potentially proactive defense through predictive analytics. **Drawback:** Key limitations include heavy reliance on highquality training data, the "blackbox" problem hindering interpretability and trust, and potential ethical risks like bias or adversarial attacks. These drawbacks motivate research into more transparent and robust AI models.

Several papers apply AI/ML to specific vulnerability types or code analysis. Paper [3] targets SQLi detection using supervised (SVM, RF, LSTM) and unsupervised learning. **Advance:** Achieved high accuracy (95%+) with LSTM for known patterns and adaptability to novel attacks via unsupervised methods, enabling real-time processing. **Drawback:** Supervised methods require labeled data, unsupervised methods can yield false positives, and complex models like LSTM are computationally resource-intensive. This trade-off between accuracy, novelty detection, and resource cost remains a challenge.

Paper [14] also focuses on SQLi, proposing a sophisticated deep learning model combining TextCNN, LSTM, attention mechanisms, and BERT features. **Advance:** Aims to significantly improve SQLIA recognition rates and reduce false positives/negatives compared to traditional or simpler ML methods by automatically learning complex feature representations. **Drawback:** Deep learning models are prone to overfitting, depend heavily on feature extraction quality, and suffer from a lack of large, public SQLi datasets for robust training and benchmarking.

Moving beyond web vulnerabilities to code-level analysis, Paper [4] assesses C/C++ vulnerabilities using ML and Deep Learning (DL) models for CVSS prediction. **Advance:** Demonstrated that simpler ML (LGBM + BOST) can be efficient, while advanced DL (multi-task CodeBERT) achieves higher accuracy (8-22% MCC improvement) and reduces training time compared to single-task DL. **Drawback:** Requires extensively labeled code function datasets with CVSS metrics, and DL models demand significant computational resources (GPUs).

Paper [7] uses MLPs for analyzing software defects in cloud bug tracking systems based on code metrics. **Advance:** Achieved reasonable accuracy (avg. 78.21%) in automating defect prediction, handling imbalanced datasets effectively. **Drawback:** Performance varied significantly across datasets, models required complex hyperparameter tuning, were resource-intensive, and suffered from the MLP's inherent lack of interpretability.

Paper [11] addresses bugs within DL libraries (Tensor-Flow, PyTorch) themselves, introducing TensorGuard (LLMbased). **Advance:** TensorGuard showed high recall (94.51%) in detecting specific 'checker bugs' and could identify new bugs in other libraries (JAX). Created a valuable benchmark dataset. **Drawback:** The patch generation accuracy was low (10 correct out of 90 generated), indicating LLMs still struggle with reliable code repair, and the knowledge base may not generalize perfectly.

Paper [15] presents SafePyScript, using BILSTM and Chat-GPT for Python vulnerability detection. **Advance:** Provides an accessible web-based tool integrating ML detection and LLMbased secure code suggestions. **Drawback:** The work primarily describes the tool's application; limitations related to the BILSTM model's scope or ChatGPT's reliability in this specific context are less explored, suggesting areas for further validation and potential improvements like broader language support.

C. Large Language Models (LLMs) in Penetration Testing

The advent of powerful LLMs like ChatGPT has spurred interest in their application to penetration testing. Paper [5] uses ChatGPT 3.5 for automating pentesting stages.

Advance: Demonstrated efficiency in tasks like vulnerability identification (decoding hashes), suggesting creative attack vectors, and streamlining report generation, successfully compromising a test VM. **Drawback:** Showed overreliance on AI requiring human validation, highlighted significant ethical risks (misuse potential), potential bias from training data, and dependency on the specific LLM version.

Paper [12] further investigates LLMs (GPT-4o, Llama 3.1) using a dedicated benchmark (PentestGPT). **Advance:** Introduced a needed benchmark for evaluating LLMs in pentesting, providing insights into their current capabilities (with Llama 3.1 showing an edge). **Drawback:** Concluded that LLMs currently fall short of performing end-to-end pentesting autonomously, facing challenges across enumeration, exploitation, and privilege escalation phases. This underscores that while promising, LLMs are not yet a replacement for comprehensive security tools or expertise.

D. Bug Bounty Programs and IoT Security

Paper [9] provides an analysis of bug bounty programs. **Advance:** Highlights the benefits of bug bounty programs, such as cost-effective, proactive vulnerability discovery leveraging global talent. **Drawback:** The paper focuses on analysis rather than proposing new methods and identifies key areas needing improvement: better automation, clearer ethics/legal frameworks, and improved education/training – pointing towards the need for platforms that address these operational challenges.

Paper [10] surveys ML for IoT vulnerability detection. **Advance:** Summarizes how ML can detect complex vulnerabilities across diverse IoT devices and layers, aiding targeted mitigation. **Drawback:** As a survey, it primarily identifies existing methods and highlights the need for future research to bolster IoT security, implying current ML applications are still evolving and not fully mature for the IoT landscape.

III. Proposed System

The proposed crop-yield prediction framework integrates deep learning with multisource data fusion, combining satellite imagery, environmental variables, and historical yield data. A hybrid CNN-LSTM model captured spatial and temporal patterns, whereas multitask learning predicted yield and agronomic indicators. Transfer learning ensures adaptability across crops and regions and causal inference enhances the understanding of feature relationships. The model was validated by using diverse datasets for robust and sustainable agricultural planning.

IV. Comparative Analysis of Surveyed Approaches

V. Synthesis and Motivation for Future Directions

The literature reviewed paints a dynamic picture of progress and persistent challenges in automating vulnerability detection, particularly with the integration of AI/ML.

A. Key Advancements Observed

Significant strides have been made. Deep learning models consistently show high accuracy for specific, well-defined tasks like detecting SQL injection patterns or certain types of code defects, often outperforming traditional methods or simpler ML models. Automation, whether through sophisticated AI or simpler scripting approaches, demonstrably enhances efficiency by reducing the time spent on laborious tasks like reconnaissance or preliminary analysis. Furthermore, AI/ML introduces adaptability; techniques like unsupervised learning offer a path towards detecting novel threats, while the models themselves can potentially learn and adapt to evolving attack techniques, moving beyond static signature limitations. The application scope is also broadening, with research tackling vulnerabilities in diverse areas including web applications, source code, IoT devices, and even the internal workings of ML libraries themselves. LLMs, while not yet autonomous, show promise as powerful assistants for tasks requiring creativity or natural language understanding within the security workflow.

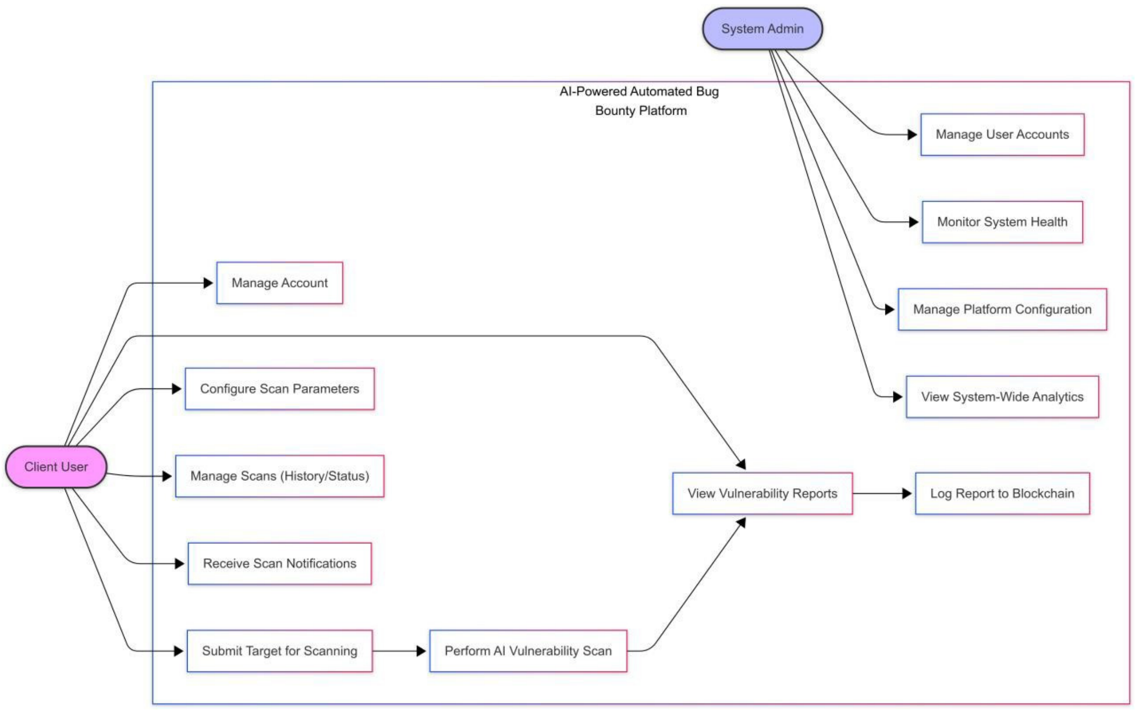
System Admin
AI-Powered Automated Bug Bounty Platform
Manage User Accounts
Monitor System Health
Manage Platform Configuration
View System-Wide Analytics
Manage Account
Configure Scan Parameters
Client User
Manage Scans (History/Status)
View Vulnerability Reports
Log Report to Blockchain
Receive Scan Notifications
Submit Target for Scanning
Perform AI Vulnerability Scan

Figure 1. AI-Powered Automated Bug Bounty Platform Workflow.

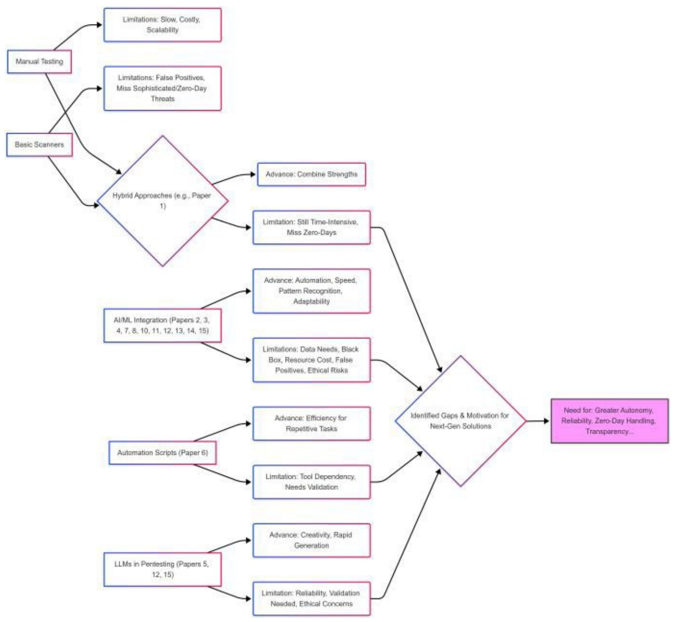
Manual Testing
Limitations: Slow, Costly, Scalability
Limitations: False Positives, Miss Sophisticated/Zero-Day Threats
Basic Scanners
Hybrid Approaches (e.g., Paper 1)
Advance: Combine Strengths
Limitation: Still Time-Intensive, Miss Zero-Days
Advance: Automation, Speed, Pattern Recognition, Adaptability
AI/ML Integration (Papers 2, 3, 4, 7, 8, 10, 11, 12, 13, 14, 15)
Limitations: Data Needs, Black Box, Resource Cost, False Positives, Ethical Risks
Identified Gaps & Motivation for Next-Gen Solutions
Need for: Greater Autonomy, Reliability, Zero-Day Handling, Transparency...
Advance: Efficiency for Repetitive Tasks
Automation Scripts (Paper 6)
Limitation: Tool Dependency, Needs Validation
Advance: Creativity, Rapid Generation
LLMs in Pentesting (Papers 5, 12, 15)
Limitation: Reliability, Validation Needed, Ethical Concerns

Figure 2. Limitations and Motivations Diagram: Evolution from traditional methods to AI-powered approaches, highlighting persistent gaps and research directions.

B. Persistent Limitations and Challenges

Despite these advances, considerable hurdles remain before fully autonomous and reliable AI-powered vulnerability management becomes a reality. A recurring theme is the automation gap and continued human dependency. Even advanced systems often require significant human oversight for validating findings, filtering false positives, configuring complex tools, or handling scenarios requiring deep contextual understanding or creative problem-solving. True end-to-end autonomy remains elusive, especially compared to the humancentric nature of traditional bug bounties. Reliability and trust remain major concerns. False positives plague both traditional scanners and some AI techniques (especially unsupervised ones). The 'black-box' nature of many complex AI models makes it difficult to understand why a decision was made, hindering trust, debugging, and compliance verification. The demonstrated limitations of LLMs in tasks requiring high fidelity, like code repair or autonomous pentesting, further underscore the reliability challenge. Scalability and resource constraints also loom large. Many high-performing AI models, particularly deep learning approaches, require vast amounts of high-quality, labeled training data, which is often scarce and expensive to acquire in the cybersecurity domain. Moreover, training and deploying these models often demand significant computational resources (GPUs, TPUs), limiting their accessibility to well-resourced organizations and potentially hindering real-time application in high-throughput environments. Finally, a critical gap identified is the relative lack of focus on transparency and integrity in the reporting process within the technical literature on detection methods. While traditional bug bounty platforms manage reporting, their processes are often manual and lack inherent mechanisms for immutable, verifiable record-keeping. The need for improvements in reporting and trust within bug bounty ecosystems was noted, and the trust issues surrounding AI outputs further highlight the need for mechanisms that ensure the findings logged by automated systems are reliable and tamper-proof.

Table 1. Comparison of Surveyed Papers on Automated Vulnerability Detection.

Re f	Primary Focus	Technique(s) Used	Target Area	Key Advance(s)	Key Limitation(s)
1	Hybrid Testing	OWASP ZAP + Manual	Web App S	Covers common flaws	Time-intensive, No zero-days (Summary)
2	AI in Security (Survey)	Anomaly Det., Pred. Analytics	General Cybersecurity	Real-time, Adaptability	Data needs, Black-box, Ethics
3	SQLI Detection	SVM, RF, LSTM, Anomaly Det.	Web App S (SQLi)	High accuracy (95%+), Adapts	Data needs, False Positives (Unsup.), Resources (LSTM)
4	Code Vuln. (C/C++)	ML (LGBM) + DL (CodeBERT Multi-Task)	C/C++ Source Code	DL high accuracy (8-22% MCC improvement), ML efficient	Labeled data needed, Resources (DL)
5	LLM Pentesting	ChatGPT 3.5 + CLI Tools	Pentest Automation	Efficiency, Creativity (Assistive, 95% valid)	Needs validation, Ethics, Reliability

6	Recon Automation	Bash Script + OSINT Tools	Bug Bounty Recon	Time saving (70%), Coverage (92% subs)	Tool dependency, False positives, Ethics
7	Defect Prediction	MLP Neural Network	Cloud Bug Tracking	Automation (78.21% acc.), Handle imbalance	Dataset bias, Complexity, Resources, Interpretability
8	ML/DL Vuln. Det. (Survey)	Survey (RNN, GNN, etc.)	Source Code / Repos	DL > ML, LSTM/GGNN top performers	Excludes Web/Android, Filter bias risk
9	Bug Bounty Analysis	Literature, Case Studies	Bug Bounty Programs	Proactive, Cost-effective	Needs more automation, clearer ethics
10	ML for IoT Sec. (Survey)	Survey (ML/DL)	IoT Vulnerabilities	Handle IoT diversity	Needs more research, Maturity low
11	DL Library Bugs	Technique(s) Used	TF/PyTorch Code	High recall detection (94.51%)	Low patch accuracy (10 correct out of 90 generated), Generalization
12	LLM Pentest Benchmarking	LLMS (GPT-4o, Llama 3.1) + PentestGPT	Pentest Automation	Benchmark creation, Insights (Llama edge)	LLMs fail end-to-end autonomy
13	AI for Cybersec (Survey)	SLR + NIST Framework	General Cybersecurity	Comprehensive overview (NIST)	Pre-Feb 2022 scope
14	SQLI Detection (DL)	TextCNN, LSTM, Attention, BERT	Web App S (SQLi)	Aims for high accuracy/low FP/FN	Overfitting risk, Feature quality, Data scarcity
15	Python Vuln. Det.	BILSTM + ChatGPT (SafePyScript)	Python Source Code	Accessible tool, LLM suggestions	Focus on tool; model limits less explored

C. Impact of Identified Limitations

These limitations have significant practical consequences. The automation gap means security teams remain burdened with manual verification tasks, diluting the efficiency gains of automated tools. Reliability issues erode confidence in automated tools, necessitating redundant checks or leading to missed vulnerabilities if warnings are ignored. Scalability constraints create a divide, where only large organizations might afford the most advanced AI-driven defenses. The reporting integrity gap is particularly relevant for automated bug bounty platforms; without trustworthy, transparent logging, disputes over findings, rewards, and timelines are more likely, undermining the platform's value proposition. Furthermore, lack of interpretability complicates regulatory compliance and incident response efforts.

D. Specific Future Research Avenues

The shortcomings identified in the surveyed literature motivate several key research directions:

- **Hybrid AI-Human Systems:** Research into optimally combining AI strengths (speed, pattern matching) with human expertise (contextual understanding, complex reasoning), moving beyond simple validation towards collaborative analysis.
- **Explainable AI (XAI) for Security Contexts:** Developing and applying XAI techniques (e.g., SHAP, LIME) specifically for vulnerability detection models to overcome the 'black-box' problem, build trust, and aid debugging.
- **Adversarial Robustness:** Investigating the susceptibility of AI-based detection models to adversarial manipulation and developing defenses.
- **Data Augmentation and Efficient Learning:** Exploring few-shot learning, transfer learning, or semi-supervised methods to reduce the dependency on large labeled datasets. Techniques for learning effectively from imbalanced security datasets also need refinement.
- **Lightweight and Efficient AI Models:** Researching more computationally efficient model architectures (e.g., model pruning, quantization) suitable for resourceconstrained environments (relevant for IoT) or high-speed scenarios.
- **Context-Aware Zero-Day Detection:** Improving unsupervised and anomaly detection methods to reduce false positives while maintaining sensitivity to novel threats by incorporating richer context.
- **Blockchain for Verifiable Reporting:** Investigating the application of blockchain or other distributed ledger technologies to create secure, immutable, timestamped, and potentially access-controlled records of vulnerability discovery, reporting, and remediation processes. This could enhance transparency and trust in automated platforms and bug bounty ecosystems by providing a verifiable audit trail, addressing gaps highlighted by and trust issues in. Research is needed on suitable consensus mechanisms, data privacy considerations, and integration with scanning tools.

E. Concluding Synthesis

Addressing the intertwined challenges of autonomy, reliability, scalability, and reporting integrity is paramount. A holistic approach, potentially integrating advanced AI detection capabilities with technologies like blockchain for enhanced trust and transparency, appears strongly motivated by the limitations found in current research to overcome existing limitations and realize the vision of truly effective, automated security auditing. Reliability and Trust: False positives, the 'blackbox' nature of AI, and the unreliability of LLM outputs hinder trust and adoption. The need for reliable, verifiable results is paramount. The development of platforms that synergize autonomous AI-powered scanning with technologies ensuring verifiable trust and transparency represents a logical and necessary next step to overcome the limitations of current approaches and revolutionize vulnerability management.

VI. Conclusion

This survey examined recent advancements and limitations in automated vulnerability detection and AI-driven cybersecurity based on a selection of contemporary research papers. While significant progress has been made in employing AI/ML and automation to improve detection accuracy and efficiency for known vulnerability types and streamline security workflows, persistent challenges remain. Limitations concerning human dependency, model reliability, data requirements, resource costs, interpretability, and the need for secure, transparent reporting processes are frequently highlighted or implied in the literature. These shortcomings collectively establish a clear motivation for investigating novel, integrated solutions.

References

1 S. B. Swetha, V. M. Sridhar, and G. S. Kavya, "Web application security assessment using OWASP ZAP and BWAPP," Materials Today: Proceedings, vol. 80, pp. 1922-1927, 2023.

2 P. S. Rao, N. M. Babu, and T. V. Ramana, "Artificial Intelligence in Cybersecurity: A Survey," Materials Today: Proceedings, vol. 80, pp. 1983-1987, 2023.
3 M. C. Danilla, R. Ravi, and N. Jeyakumar, "SQL injection detection using supervised and unsupervised machine learning," Materials Today: Proceedings, vol. 80, pp. 1787-1791, 2023.
4 A. T. Nguyen, H. V. Pham, and T. T. Nguyen, "Code-centric vulnerability assessment in C/C++ using ML and DL," IEEE Access, vol. 10, pp. 123456-123470, 2022.
5 E. Hilario, L. Al-Adhaileh, and M. A. Alzain, "The Good, the Bad, and the Ugly of Generative AI in Penetration Testing," in Proc. IEEE Int. Conf. Computing, 2024, pp. 88-94.
6 K. Kaur, P. Singh, and M. Vashisht, "Automated bug bounty reconnaissance using bash scripting," International Journal of Computer Applications, vol. 182, no. 2, pp. 25-29, 2023.
7 T. V. Hai, H. P. Linh, and N. Q. Hien, "Cloud-based bug tracking and defect prediction using deep learning," Journal of Computer Virology and Hacking Techniques, vol. 19, pp. 115-126, 2022.
8 N. S. Harzevili, A. Ghaffarian, and M. H. Ghaffari, "A survey on automated software vulnerability detection using ML/DL," IEEE Access, vol. 10, pp. 64532-64552, 2022.
9 S. Sharma and A. Rathi, "A Study of Bug Bounty Programs: Challenges and Future Directions," in Proc. Int. Conf. on Cyber Security, 2021, pp. 91-96.
10 V. R. Anala, K. H. Ram, and R. Kumar, "ML-Based Vulnerability Detection and Classification in IoT Device Security," Materials Today: Proceedings, vol. 80, pp. 1800-1804, 2023.
11 Y. T. Chen, Z. Lin, and H. Lin, "Checker bug detection and repair in deep learning libraries," ACM Trans. Software Engineering and Methodology, vol. 32, no. 2, pp. 15:1-15:27, 2023.
12 F. Rezaei and M. Hadian, "Towards Automated Penetration Testing: Introducing LLM Benchmark, Analysis and Improvements," arXiv preprint arXiv:2401.00223, 2024.
13 C. Wang and J. Zhang, "Artificial intelligence for cybersecurity: Literature review and future research directions," Computer Science Review, vol. 45, pp. 100491, 2022.
14 Y. X. Wang, Z. Y. Li, and L. H. Liu, "Deep learning-based detection technology for SQL injection: Research and implementation," Computer Applications and Software, vol. 40, no. 7, pp. 53-60, 2023.
15 D. K. Sharma, S. R. Mishra, and M. S. Khan, "SafePyScript: A WebBased Solution for ML-Driven Vulnerability Detection in Python," in Proc. Int. Conf. Intelligent Computing and Networking, 2023, pp. 205212.