*European Journal of AI, Computing & Informatics*

*Vol. 1 No. 3 2025*

*Aeticle* **Open Access**

# Application of Network Security Vulnerability Detection and Repair Process Optimization in Software Development

**Shuang Yuan** [1,*]

[1] Technology Risk Management, American Airlines, Fort Worth, Texas, 76155, United States
* Correspondence: Shuang Yuan, Technology Risk Management, American Airlines, Fort Worth, Texas, 76155, United States


Received: 29 August 2025
Revised: 18 September 2025
Accepted: 29 October 2025
Published: 31 October 2025

**Copyright:** © 2025 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

**Abstract:** Ensuring system safety and maintaining high-quality software development critically depend on the timely identification and rapid remediation of network security vulnerabilities. In this study, an efficient operating framework is established based on the optimization of the vulnerability management process. From the dual perspectives of vulnerability detection and repair, a comprehensive process improvement is implemented, forming a technical framework that integrates both detection and repair optimization modules. This framework emphasizes security robustness while offering flexible scalability, enabling it to adapt to varying software environments and evolving threat landscapes. During the software development lifecycle, the framework leverages the coordinated application of multiple tools, including automated repair technologies, intelligent detection mechanisms, and cross-department collaboration strategies. Such integration significantly enhances both the efficiency and accuracy of handling security vulnerabilities. Furthermore, the proposed approach supports continuous monitoring and iterative refinement, ensuring that potential security risks are proactively addressed, and that the overall reliability and stability of software systems are improved. By systematically combining detection, repair, and collaborative optimization, the framework provides a practical and scalable solution for strengthening software security and supporting sustainable software development.

**Keywords:** network security; vulnerability detection; bug repair; repair process; software development

## 1. Introduction

In today's era of rapid digitization and pervasive information technology, software security plays a critical role in shaping both organizational operations and societal outcomes. As software systems grow in complexity and scale, the types and severity of network security vulnerabilities are simultaneously increasing. These vulnerabilities not only threaten the integrity and confidentiality of data but also pose risks to system availability, business continuity, and user trust.

Traditional approaches to managing security vulnerabilities typically rely on manual detection, isolated patching, and reactive remediation. While these methods can address specific issues, they often suffer from inefficiencies, human error, and the inability to capture complex or latent vulnerabilities. In large-scale software projects or environments following agile and continuous integration/continuous deployment (CI/CD) methodologies, the limitations of manual vulnerability management are particularly pronounced. Teams may overlook critical security risks, resulting in potential breaches or operational failures that could have been prevented through proactive strategies [1].

To address these challenges, modern approaches increasingly emphasize automation, intelligence, and collaborative processes. Automated vulnerability detection tools can continuously scan codebases, network environments, and system configurations to identify potential risks in real-time. Machine learning and artificial intelligence techniques further enhance detection by analyzing patterns, predicting emerging threats, and prioritizing remediation based on risk impact. Meanwhile, collaborative platforms allow multiple stakeholders-developers, security engineers, and operations personnel-to coordinate vulnerability management throughout the entire software lifecycle, from discovery to verification and patch deployment.

By integrating automated, intelligent, and collaborative technologies, organizations can achieve a more comprehensive and proactive approach to security management. This approach not only improves the efficiency and quality of vulnerability detection and remediation but also strengthens overall system resilience. Moreover, as digital ecosystems expand to include cloud services, Internet of Things (IoT) devices, and interconnected operational technologies, such holistic strategies become essential for maintaining trust, ensuring compliance, and minimizing operational disruptions [2].

In summary, the evolution from reactive, manual security practices toward automated, data-driven, and collaborative vulnerability management represents a necessary transformation for modern software development. By embracing these strategies, organizations can safeguard complex digital infrastructures while supporting agile innovation and sustainable growth.

## 2. Optimization Method of Network Security Vulnerability Detection and Repair Process

### *2.1. Optimization Process of Vulnerability Detection*

Optimizing the vulnerability detection process requires balancing accuracy, efficiency, and coverage. A comprehensive approach combines static and dynamic analysis methods. Static code review allows for the rapid identification of potential defects in programming logic, code structure, and security patterns, while dynamic testing monitors system behavior and risk points during execution, capturing vulnerabilities that only emerge at runtime.

Artificial intelligence, particularly machine learning techniques, can further enhance detection accuracy by predicting the likelihood and severity of potential vulnerabilities based on historical data and code patterns. Integrating automated code scanning, penetration testing, and modular analysis pipelines allows the system to progressively detect and classify vulnerabilities, significantly reducing the reliance on manual inspection.

To meet the demands of modern software development, especially agile and CI/CD environments, the vulnerability detection process should evolve toward real-time monitoring. Continuous integration of scanning and testing tools enables developers to receive immediate feedback on security risks as code is written and deployed. This approach not only improves the comprehensiveness and reliability of detection but also significantly accelerates the inspection cycle, providing a dual benefit of higher efficiency and enhanced accuracy [3].

Furthermore, intelligent orchestration of detection tasks allows resources to be dynamically allocated based on system complexity and workload, ensuring that critical modules receive priority attention. By implementing such an optimized detection workflow, organizations can establish a proactive security posture, catching vulnerabilities early and reducing the likelihood of exploitation in production systems.

### *2.2. Optimization Process of Vulnerability Repair*

Effective and efficient repair of security vulnerabilities is essential to maintaining software quality and system reliability. Traditional repair processes often involve long

cycles, high costs, and uncertainty regarding the impact of fixes on other system functions. Optimizing this process requires establishing an automated, standardized, and intelligent repair workflow [4].

Prioritization of vulnerabilities is a critical step. Automated security assessment tools can classify vulnerabilities according to risk level, exploitability, and urgency, enabling teams to focus their efforts on high-priority issues. Intelligent technologies can assist in generating repair code suggestions, recommending remediation strategies, or proposing standardized repair templates, reducing human error and improving consistency across development teams.

Collaboration is also central to optimized repair processes. Automated task assignment ensures that repair responsibilities are clearly distributed, while integration with CI/CD pipelines allows for rapid verification and deployment of fixes. Real-time monitoring and feedback mechanisms verify the effectiveness of repairs and detect any regression or unintended side effects, ensuring the stability of the system [5,6].

Additionally, repair optimization benefits from continuous learning. Data from past vulnerabilities, repair outcomes, and incident response metrics can be fed into machine learning models to improve the predictive accuracy of risk assessment and remediation guidance. This creates a self-improving repair ecosystem where both detection and remediation processes are increasingly efficient, intelligent, and adaptive to evolving security threats.

By implementing a fully optimized detection and repair framework, organizations can achieve a robust, end-to-end approach to software security management. Such a framework ensures that vulnerabilities are identified and resolved promptly, supports rapid development cycles, and establishes a resilient foundation for secure software operations in complex digital environments.

## 3. Network Security Vulnerability Detection and Repair Process Optimization Technical Architecture Design

### *3.1. Overall Design and Objectives of the Technical Architecture*

To enhance software security and improve operational efficiency, an improved technical framework for network security vulnerability detection and repair has been designed. The primary objective of this architecture is to accelerate the identification and resolution of vulnerabilities, thereby mitigating the potential risks they pose to the system [7].

The overall architecture consists of three main components: the vulnerability detection module, the repair optimization module, and the whole-process integration module. In the vulnerability detection module, automated scanning and artificial intelligence techniques are employed to identify vulnerabilities quickly and accurately, reducing reliance on manual inspection. The repair optimization module leverages intelligent repair program generation and standardized repair processes to ensure that remediation measures are both effective and reliable. Finally, the whole-process integration module establishes a closed-loop security management system by tightly integrating with development tools, enabling real-time monitoring, feedback, and continuous improvement throughout the software lifecycle.

The core modules, their functional characteristics, and the corresponding implementation objectives of this optimized technical architecture are summarized in Table 1 below.

**Table 1.** Overview of network security vulnerability detection and repair process optimization technical architecture design.

| Module name | Function description | Technical characteristics | Achieve the goal |
|---|---|---|---|
| Vulnerability detection module | Identify vulnerabilities using automated scanning and artificial intelligence analytics | Efficient and accurate vulnerability scanning and analysis | Quickly identify potential vulnerabilities and reduce omissions |
| Repair optimization module | Optimize the vulnerability repair process by automatically generating and validating fix patches | Intelligent patch generation and verification mechanism | Ensure the effectiveness and reliability of repair measures |
| Whole process integration module | Seamless integration with the development tool chain to achieve end-to-end closed-loop security management | Integrated development and safety management system | Improved vulnerability handling |

The primary goal of this architecture is to reduce the cycle time for detecting and resolving security vulnerabilities while simultaneously improving the quality of remediation. This ensures that the software development process remains both secure and efficient, minimizing potential risks and supporting continuous, high-quality code delivery.

### *3.2. Construction of the Technical Architecture of Vulnerability Detection Module*

The vulnerability detection module serves a critical role within the network security system, and its technical architecture must carefully balance response speed, detection accuracy, and scalability. Built on a multi-level detection mechanism, the module integrates static source code analysis, dynamic behavior monitoring, and rule-based risk model construction to provide comprehensive oversight of potential security vulnerabilities throughout all stages of the software development life cycle [8].

Leveraging a microservice architecture, each functional component operates independently, while standardized interfaces facilitate smooth interaction between modules. This design enhances the system's flexibility, maintainability, and adaptability to evolving security requirements. Advanced artificial intelligence techniques, particularly machine learning, are employed to analyze historical vulnerability data and train a self-adjusting detection engine. This enables the module to efficiently detect new and complex vulnerabilities that may not be captured by conventional methods [9].

In addition, the system incorporates efficient data acquisition and analysis tools, ensuring the comprehensive collection and timely reporting of vulnerability information. Real-time monitoring logs and data streams are aggregated and analyzed to provide actionable insights, enabling rapid identification and prioritization of high-risk vulnerabilities. Table 2 below illustrates the four-layer design logic of the technical architecture for the vulnerability detection module, highlighting its modular, intelligent, and data-driven structure.

**Table 2.** Overview of the four-layer design and functions of the technical architecture of the vulnerability detection module.

| hierarchy | Core function | Technical realization | advantage |
|---|---|---|---|
| Data acquisition layer | Collect vulnerability related data, such as static code, run logs, etc | Static code analysis tools, log collectors, API calls | Provide a comprehensive data |

| | | | |
|---|---|---|---|
| | | | source to ensure the basis of testing |
| Detection and analysis layer | Vulnerability identification, including static analysis, dynamic monitoring, and threat modeling | Rule matching engine, machine learning model, behavior monitoring module | Improve the accuracy and coverage of vulnerability identification |
| Decision response layer | The detection results are classified, prioritized and responded to | Automatic hierarchical algorithm, response strategy engine | Ensure fast handling of high-priority security threats |
| Integrated display layer | Display test results, generate reports, and support development process optimization | Dashboards, report generation tools, integration with CI/CD systems | Improve information visualization and decision-making efficiency |

The vulnerability detection module combines a multi-level technical architecture with advanced detection techniques to achieve comprehensive collection and accurate evaluation of vulnerability information. It further strengthens system security and operational performance by providing rapid decision feedback and clear, actionable intelligence.

### *3.3. Construction of Technical Architecture of Repair Process Optimization Module*

The technical architecture of the repair process optimization module emphasizes automation and intelligence to enhance both the efficiency and accuracy of vulnerability remediation. Leveraging the security event analysis component, the module captures and analyzes real-time data returned from the vulnerability detection module, classifies the threat level of each vulnerability, and determines the appropriate processing sequence. Integrated with a rule-based automatic patch generation tool and an AI-assisted code repair recommendation system, the module formulates tailored repair strategies. Subsequently, the test validation component evaluates the feasibility and system compatibility of the proposed repair strategies, mitigating the risk of introducing new vulnerabilities during the remediation process. Finally, the repair deployment scheduler pushes the validated patches to the relevant systems and monitors their effectiveness, creating a closed-loop feedback mechanism that supports dynamic optimization. Figure 1 below illustrates the complete technical architecture of the repair process optimization module [10].

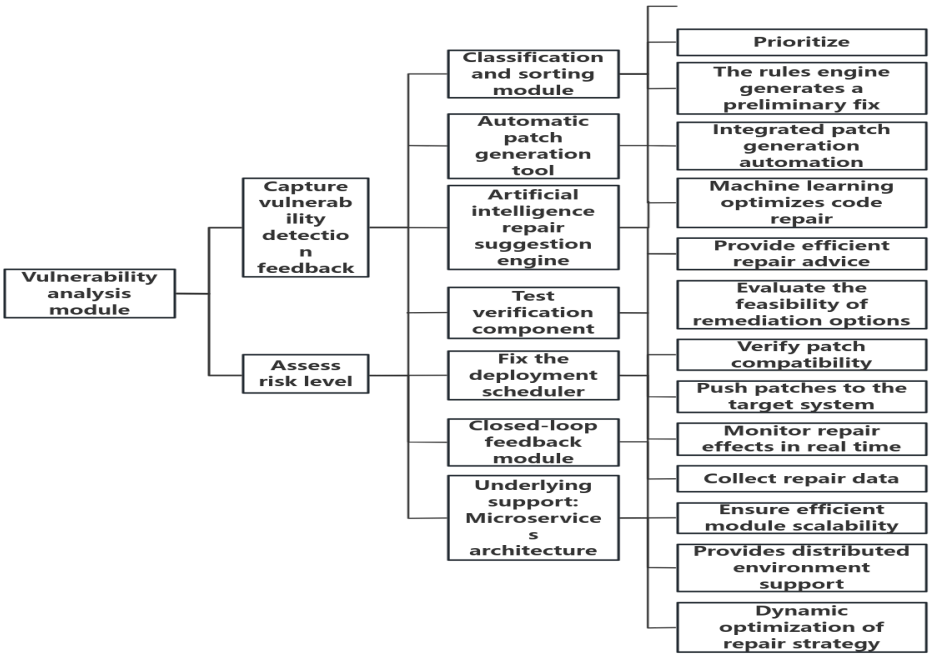


**Figure 1.** Organizational structure of repair process optimization module.

The entire module architecture is deployed using a microservice approach to ensure high scalability and stability, enabling efficient operation in large-scale distributed environments.

### *3.4. Security and Scalability of Technical Architecture*

The security and scalability of the technical architecture are critical for optimizing the network security vulnerability detection and repair process. To achieve robust security, the design employs a layered defense strategy, providing comprehensive protection across the network, application, and data levels. Unauthorized access is strictly controlled, and user authentication mechanisms are strengthened to ensure secure internal system communications. Additionally, the architecture integrates a real-time threat detection and response system that leverages machine learning to dynamically analyze data traffic and logs, enabling rapid identification and mitigation of potential risks.

To enhance scalability, the system adopts a microservice architecture, modularizing the detection and repair functions. Each service can be deployed and scaled independently, allowing resources to be adjusted according to workload. Containerization technologies, such as Docker, combined with automated orchestration tools like Kubernetes, enable rapid cross-platform deployment and streamlined application management. The architecture also supports seamless integration with multi-cloud platforms, effectively eliminating single points of failure and improving system reliability and adaptability [11].

This design not only ensures robust security but also delivers high scalability, allowing the system to respond efficiently to evolving security challenges and dynamic software development environments. It establishes a solid technical foundation for sustainable enterprise operations while facilitating continuous improvement in vulnerability detection and repair capabilities.

## 4. Application of Optimization of Security Vulnerability Detection and Repair Process in Software Development

### *4.1. Application of Multi-Tool Collaboration in Development*

In modern software development, strengthening the optimization of security vulnerability detection and repair processes is essential to ensure software quality and protect user information. Multi-tool collaboration plays a critical role in achieving this goal. Traditional single-tool approaches are often insufficient, as they cannot comprehensively identify all potential risks. By integrating multiple tools-such as static code analysis tools (e.g., SonarQube), dynamic vulnerability scanning tools (e.g., OWASP ZAP), and dependency library security monitoring tools (e.g., Snyk)-a comprehensive, multi-dimensional vulnerability detection network can be established.

This collaborative tool ecosystem can be seamlessly incorporated into continuous integration/continuous deployment (CI/CD) pipelines throughout the software development lifecycle. Static analysis tools are automatically triggered at code submission to detect coding errors and potential security risks. During deployment, dynamic analysis tools simulate attack scenarios and uncover runtime vulnerabilities, while dependency monitoring tools detect security issues within third-party libraries and components. The integration of these tools enhances the comprehensiveness of vulnerability detection and reduces false positives through shared information and cross-validation among tools.

During the vulnerability repair phase, multi-tool collaboration remains equally important. By combining a vulnerability management platform with development management tools (e.g., JIRA), teams can rapidly identify, assign, and track repair tasks. Automated patch generation and AI-assisted repair suggestions further accelerate the remediation process, significantly reducing the time required to resolve high-priority vulnerabilities [5].

Overall, multi-tool collaboration not only improves the breadth and depth of vulnerability detection but also shortens repair cycles, enhances team efficiency, and strengthens coordination across development and security teams. It has become a core strategy for optimizing the security processes in modern software development, ensuring both resilience and efficiency throughout the software lifecycle.

### *4.2. Application of Automated Repair Technology*

In the software development process, automated repair technology can rapidly identify and address security risks through intelligent operations and efficient workflows. This approach significantly improves the efficiency of vulnerability remediation while reducing the likelihood of security breaches during development. The core steps of this technology include vulnerability identification, automatic patch generation, and validation of patch effectiveness.

Based on the results of vulnerability scanning, an automated repair strategy is formulated according to the specific characteristics and severity of each vulnerability. The system then selects the optimal repair solution and applies the corresponding patch. This process ensures that high-risk vulnerabilities are prioritized and addressed promptly, while minimizing disruption to system functionality. The optimal selection of repair strategy can be represented through a formulaic approach that considers factors such as risk level, repair cost, and expected impact on system performance, enabling the most efficient and effective remediation decisions [6].

$$s^* = arg \max_{s \in S} Effectiveness(s, D) \quad (1)$$

Among them, Effectiveness$(s, D)$Indicates the effect of the repair policy on a specific vulnerability. Patches generate repaired code based on program context analysis and predefined templates$C'$The functional correctness and security verification conditions after vulnerability repair must be met. Functional correctness verification formula:

$$\forall x \in TestCases, Exec(C', x) = ExpectedOutput(x) \quad (2)$$

Among them, $\mathrm{TestCases}$ For a collection of test cases, $\mathrm{Exec}$Indicates the execution result of the code. Technologies such as deep learning continue to develop, and repair algorithms based on a large amount of data continue to evolve, which can update repair schemes in real time according to past vulnerability repair cases, thus greatly improving the wisdom of the automatic repair system and providing strong support for the security of the programming process.

*4.3. Application of Intelligent Detection Technology*

In software development, intelligent detection technology relies on deep learning and natural language processing (NLP) technology, which significantly improves the efficiency of detecting and fixing security vulnerabilities. This kind of technology can automatically screen out key features in a large number of code, and through model training to predict new vulnerabilities, providing powerful technical support for developers. In the intelligent detection process, code semantic analysis is the core link. By converting code into vector data and feeding it into a deep learning model, the system can automatically detect potential vulnerabilities. Let the semantic vector of the code snippet be $\mathrm{C} = [\mathrm{c}_1, \mathrm{c}_2, \cdots, \mathrm{c}_\mathrm{n}]$, intelligent detection model by building a scoring function$\mathrm{S(C, W)}$To evaluate the security of the code, which$\mathrm{W}$Is the model parameter. Code scoring function formula:

$$S(C, W) = \sum_{i=1}^{n} w_i \cdot c_i \quad (3)$$

If score $\mathrm{S(C, W)}$Less than the safety threshold, is flagged as potentially vulnerable code. Intelligent detection technology is also widely used in dynamic behavior analysis to capture abnormal behavior by simulating the runtime environment. Set the sequence of logs generated when a piece of code is run $\mathrm{L} = [l_1, l_2, \cdots, l_m]$. Abnormal score formula:

$$E(L) = \frac{1}{m}\sum_{i=1}^{m}(l_i - \mu)^2 \quad (4)$$

Among them,$\mu$ Indicates the average value of normal behavior logs. If abnormal score $\mathrm{E(L)}$ If the threshold is exceeded, the system triggers a security alarm. Advanced technical intelligence not only realizes vulnerability detection, but also optimizes and upgrades the patching process. With continuous learning, models can be adjusted in real time to address emerging vulnerabilities.

*4.4. Optimization of Cross-Team Collaboration and Security Mechanism*

In traditional software development, information exchange between developers and security personnel is often inefficient, which can delay the detection and remediation of security vulnerabilities and, in some cases, exacerbate security risks. To address this challenge, optimizing cross-team collaboration mechanisms and strengthening security protocols are critical measures. Integrating security considerations throughout every stage of software development enhances the capacity for early detection and rapid resolution of vulnerabilities.

The adoption of DevSecOps practices facilitates close collaboration between development and security teams from the outset of a project. By leveraging advanced techniques such as static code analysis and dynamic testing tools, teams can accurately identify and mitigate potential security risks. Automated testing systems further streamline the detection process and reduce the need for manual intervention, thereby improving both the speed and accuracy of vulnerability identification [8].

Effective teamwork is also reinforced through efficient communication tools and collaborative platforms, which enhance information transparency and enable immediate feedback. Real-time communication allows developers to quickly receive guidance from security specialists and adjust development activities accordingly. Additionally, the implementation of a vulnerability management system provides full visibility into the repair process, ensuring both the completeness and manageability of remediation efforts.

From a security mechanism perspective, establishing a repair prioritization strategy based on the criticality and potential impact of vulnerabilities promotes the rational allocation of resources and accelerates the overall repair process. By combining optimized collaboration and structured security mechanisms, software teams can achieve higher reliability, reduced risk, and more efficient resolution of security vulnerabilities.

## 5. Conclusion

The optimization of the software vulnerability detection and repair process not only enhances development speed and software quality but also mitigates security risks, ensuring the stability and protection of products in dynamic network environments. To achieve this, a series of practical optimization strategies have been proposed, along with a technical framework aligned with current software development trends.

In practical applications, the integration of multi-tool collaboration, automated repair mechanisms, and intelligent detection technologies significantly improves the timeliness and accuracy of vulnerability detection and remediation. These approaches enhance cross-team collaboration, strengthen communication efficiency between development and security teams, and provide a robust foundation for the security maintenance of the entire software life cycle.

As technology continues to evolve and these solutions are widely adopted, software security development is poised to enter a more efficient and intelligent era. This framework not only addresses present security challenges but also establishes a scalable and adaptable system capable of responding to future developments, laying the groundwork for sustainable, high-quality, and secure software solutions.

## References

1. H. Shim, J. Back, Y. Eun, G. Park, and J. Kim, "Zero-dynamics attack, variations, and countermeasures," In *Security and Resilience of Control Systems: Theory and Applications*, 2022, pp. 31-61. doi: 10.1007/978-3-030-83236-0_2
2. G. Chen, H. Wang, and C. Zhang, "Mobile cellular network security vulnerability detection using machine learning," *International Journal of Information and Communication Technology*, vol. 22, no. 3, pp. 327-341, 2023.
3. N. Hussein, and A. Nhlabatsi, "Living in the dark: Mqtt-based exploitation of iot security vulnerabilities in zigbee networks for smart lighting control," *IoT*, vol. 3, no. 4, pp. 450-472, 2022. doi: 10.3390/iot3040024
4. R. Caviglia, "Novel Approaches to Standard Based Cybersecurity Risk Management in OT Environment," 2025.
5. A. Odlyzko, "Cybersecurity is not very important," *Ubiquity*, vol. 2019, no. June, pp. 1-23, 2019. doi: 10.1145/3333611
6. N. Ziems, and S. Wu, "Security vulnerability detection using deep learning natural language processing," In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, May, 2021, pp. 1-6. doi: 10.1109/infocomwkshps51825.2021.9484500
7. X. Yang, and M. S. Mannan, "The development and application of dynamic operational risk assessment in oil/gas and chemical process industry," *Reliability Engineering & System Safety*, vol. 95, no. 7, pp. 806-815, 2010.
8. J. Roldán-Gómez, J. Carrillo-Mondéjar, J. M. Castelo Gómez, and S. Ruiz-Villafranca, "Security Analysis of the MQTT-SN Protocol for the Internet of Things," *Applied Sciences*, vol. 12, no. 21, p. 10991, 2022. doi: 10.3390/app122110991
9. J. Wetzels, D. Dos Santos, and M. Ghafari, "Insecure by design in the backbone of critical infrastructure," In *Proceedings of Cyber-Physical Systems and Internet of Things Week 2023*, 2023, pp. 7-12. doi: 10.1145/3576914.3587485
10. S. Kumar, and H. Vardhan, "Cyber security of OT networks: A tutorial and overview," *arXiv preprint arXiv:2502.14017*, 2025.
11. M. Rodda, and V. Mavroudis, "Analysis of Publicly Accessible Operational Technology and Associated Risks," *arXiv preprint arXiv:2508.02375*, 2025.

**Disclaimer/Publisher's Note:** The views, opinions, and data expressed in all publications are solely those of the individual author(s) and contributor(s) and do not necessarily reflect the views of PAP and/or the editor(s). PAP and/or the editor(s) disclaim any responsibility for any injury to individuals or damage to property arising from the ideas, methods, instructions, or products mentioned in the content.