The American Journal of Applied Sciences
ISSN 2689-0992 | Open Access

TYPE Original Research
PAGE NO. 70-76

Check for updates

Cybersecurity, Artificial Intelligence, And Iot Integration: A Multilayered Framework For Secure Intelligent Systems

Johnathan A. Roberts

Global Institute of Cybersecurity Studies, New Haven, USA

OPEN ACCESS

SUBMITTED 10 October 2025
ACCEPTED 07 November 2025
PUBLISHED 28 November 2025
VOLUME Vol.07 Issue11 2025

CITATION
Johnathan A. Roberts. (2025). Cybersecurity, Artificial Intelligence, And Iot Integration: A Multilayered Framework For Secure Intelligent Systems. The American Journal of Applied Sciences, 7(11), 70–76. Retrieved from https://www.theamericanjournals.com/index.php/tajas/article/view/6962

Abstract: Effective cybersecurity in critical infrastructure systems has become an urgent global necessity. As industries increasingly deploy continuous integration/continuous deployment (CI/CD) pipelines, surge in interconnected systems, and pervasive use of artificial intelligence (AI), vulnerabilities proliferate in both software and operational layers. This paper presents a novel, integrated framework that synthesizes established cybersecurity standards with cutting-edge AI-based vulnerability detection and demand forecasting models, aiming to secure CI/CD-powered critical infrastructure systems. Leveraging the principles of the National Institute of Standards and Technology (NIST) cybersecurity framework (NIST, 2018) as a foundational scaffold, the proposed methodology incorporates AI-driven code analysis, anomaly detection, resource demand forecasting, and continuous vulnerability management. We discuss theoretical underpinnings, detail a comprehensive methodology, and enumerate expected results. Our discussion explores potential limitations, ethical considerations, and a roadmap for future research. This work contributes to bridging the gap between standardized cybersecurity guidelines and dynamic, AI-enhanced defense in modern CI/CD and critical infrastructure environments.

Keywords: CI/CD security, AI-based vulnerability detection, NIST cybersecurity framework, demand forecasting, infrastructure resilience, code-as-graph, intrusion detection systems.

Introduction: In recent decades, digital transformation has driven critical infrastructure systems—such as energy grids, healthcare systems, supply chains, and retail ecosystems—to adopt continuous integration/continuous deployment (CI/CD) pipelines. Such pipelines allow rapid, frequent software updates,

enhancing agility but introducing new attack surfaces. Simultaneously, proliferation of Internet-of-Things (IoT) devices and AI-driven automation has woven complexity into system architectures, making them more resilient and, paradoxically, more fragile in terms of security. Thus, ensuring cybersecurity in these environments is an existential imperative.

The foundational guideline for improving security in critical infrastructure is provided by the NIST Cybersecurity Framework (NIST, 2018). The framework offers a structure of core functions—Identify, Protect, Detect, Respond, Recover—to enable organizations to manage and reduce cybersecurity risk. However, while the framework provides conceptual clarity, its implementation often faces practical challenges in modern, AI-augmented, CI/CD-heavy systems. Traditional manual audits and reactive security mechanisms struggle to keep pace with continuous deployment and rapidly evolving codebases. There is a pressing need to augment the NIST framework with automated, AI-driven tools that can operate at scale, integrate seamlessly into CI/CD pipelines, and detect vulnerabilities proactively in both code and runtime environments.

In recent years, researchers have proposed various AI-based approaches for software vulnerability detection. For example, treating source code as a graph and applying graph learning techniques to detect vulnerabilities (Suneja et al., 2020); or using deep neural networks to detect potential defects in code (Chen et al., 2020). Concurrently, AI-driven models have been used to detect anomalies and intrusions in network traffic or system behavior to flag runtime threats (Gopireddy, 2018). However, these efforts remain fragmented—many focus solely on code or network, but not both; few integrate into real-world CI/CD pipelines; and even fewer align with a comprehensive cybersecurity governance framework. Moreover, as AI becomes embedded across supply chains and retail-platform infrastructures, there is growing interest in integrating security with operational concerns like inventory management and demand forecasting (Gopireddy, 2024; Malik et al., 2025). This convergence highlights a significant gap: the absence of an integrated, standardized, AI-driven security and resource management framework for CI/CD-powered critical infrastructure.

The goal of this paper is to propose an integrated framework that merges the conceptual strengths of the NIST Cybersecurity Framework with state-of-the-art AI-driven vulnerability detection, runtime intrusion detection, and demand forecasting mechanisms. Our contributions are as follows:

1. We articulate a multi-layered architecture that embeds AI-based code vulnerability detection, static and dynamic analysis, runtime anomaly detection, and demand forecasting into a CI/CD pipeline, governed by NIST-aligned controls.

2. We provide a detailed methodology for implementing this framework in real-world CI/CD-powered systems, focusing on retail and other critical infrastructure domains.

3. We describe expected outcomes and benefits, such as reduced vulnerability exposure, faster detection and remediation, improved resource forecasting, and enhanced resilience.

4. We offer a deep theoretical discussion, acknowledging limitations, ethical implications, and outlining future research directions.

By addressing the literature gap between static standards and dynamic AI-enabled practice, our framework aims to empower organizations to safeguard critical infrastructure in an era of rapid change.

Methodology

In proposing this integrated framework, we design a methodology that combines guidelines from authoritative standards with AI-based detection and forecasting techniques, structured in a layered, modular architecture. The methodology comprises the following components: architectural design, code-level vulnerability detection, runtime intrusion and anomaly detection, demand forecasting for infrastructure resource optimization, and continuous feedback and improvement. Each component maps onto functions in the NIST framework, enabling both compliance and proactive security.

Architectural Design and Framework Mapping

We begin by designing a layered architecture whose components align with the core functions of the NIST framework: Identify, Protect, Detect, Respond, and Recover.

- Identify: We require an inventory of all assets—including software repositories, CI/CD systems, deployed services, IoT devices, data stores, user access control lists, dependencies, and infrastructure components. For CI/CD-powered systems, this includes tracking all pipelines, versions, and deployment environments.
- Protect: Implement controls such as access management, secure coding practices, code signing, encryption, network segmentation, and configuration management.
- Detect: Deploy AI-based static code analyzers,

dynamic analysis tools, runtime intrusion/anomaly detection systems, and resource demand forecasting monitors.

- Respond: Incorporate automated alerting, rollback mechanisms in CI/CD pipelines, patch deployment, and incident response procedures.
- Recover: Maintain backups, redundancy, and disaster recovery strategies; ensure restored systems are hardened, with updated security checks, and lessons from incidents feed back into Identify and Protect.

By mapping AI-driven mechanisms into this structure, we retain the rigor and comprehensiveness of a standards-based approach while leveraging automation and scalability.

Code-Level Vulnerability Detection

At the heart of our methodology is an AI-driven code vulnerability detection module integrated directly into CI/CD pipelines. This module comprises two approaches:

1. Graph-based source code analysis: Drawing inspiration from Suneja et al. (2020), we represent source code as graphs capturing control flows, data flows, abstract syntax, dependencies, and call graphs. We then apply graph neural network techniques or other graph learning algorithms to detect patterns indicative of vulnerabilities. This approach allows modeling of complex interdependencies—especially in modular or microservice architectures common in CI/CD deployments.

2. Deep Learning-based code defect detection: Building on work such as by Chen et al. (2020), we apply deep learning models (e.g., recurrent neural networks, transformers) trained on large corpora of code annotated for vulnerability and security defects. During each CI/CD commit or pull request, the module automatically analyzes new or changed code segments to identify potential vulnerabilities, insecure coding patterns, or risky dependencies.

The module flags suspicious code changes before they are merged into production, enabling developers to review, remediate, or reject potentially harmful changes.

Runtime Intrusion and Anomaly Detection

Static code analysis, while powerful, cannot detect vulnerabilities introduced at runtime—for example through misconfiguration, unexpected input patterns, novel exploits, or dynamic dependencies. Therefore, the methodology includes an AI-driven runtime detection module modeled after intrusion detection systems (IDS) and anomaly detection strategies. This draws on classical approaches (Gopireddy, 2018) but adapted for modern, AI-augmented infrastructures.

This module monitors system behavior, network traffic, user interactions, resource utilization, API calls, external connections, and other telemetry data. It leverages machine learning models—e.g., unsupervised anomaly detectors, recurrent neural networks, autoencoders, or clustering-based methods—to establish baseline behavior profiles and detect deviations indicative of potential intrusions, data exfiltration, or unauthorized access. Alerts trigger real-time notifications, automatic isolation of affected components, or rolling back to safer configurations through the CI/CD pipeline.

Demand Forecasting and Inventory Optimization for Infrastructure Resources

An often-overlooked aspect of securing CI/CD-powered critical infrastructures—particularly in retail and supply chain domains—is resource management. Work by Malik et al. (2025) demonstrates how AI-driven demand forecasting and inventory optimization can integrate with vulnerability management systems to optimize resource allocation, manage patches, and ensure capacity for rapid deployment of security fixes.

In our methodology, a demand forecasting module monitors patterns in resource usage, such as server load, storage consumption, data throughput, and other infrastructure metrics. Using time-series forecasting (e.g., multilayer neural networks, recurrent neural networks), the module predicts demand surges—potentially triggered by security updates, patches, or incident responses—and adjusts resource provisioning proactively. This ensures that security-related deployments do not cause unintended disruptions or performance bottlenecks, maintaining system resilience even during critical updates.

Continuous Feedback and Improvement

Finally, our methodology emphasizes a continuous feedback loop. Data from code-level detection, runtime monitoring, and demand forecasting feeds into periodic security audits, patch cycles, and improvements in security policies, coding guidelines, dependency management, access control, and incident response procedures. Over time, the system adapts to emerging threats, developer behavior, and usage patterns, enhancing security posture and infrastructure stability.

Throughout all modules, we enforce logging, documentation, version control, and metrics—enabling traceability, accountability, and compliance with governance requirements.

Results

Given the scope and nature of this work—as a

theoretical and conceptual proposal—the "results" are primarily projected outcomes and anticipated benefits derived from integrating the aforementioned modules. We discuss these in detail, considering both security and operational dimensions.

Reduction in Code-Level Vulnerabilities Prior to Deployment

By integrating AI-based static and dynamic code analyzers into the CI/CD pipeline, we anticipate a significant reduction in the number of vulnerabilities introduced into production. Graph-based code analysis allows detection of complex vulnerability patterns—e.g., those arising from convoluted dependencies or inter-module interactions— which traditional linters or manual reviews might miss. Deep learning-based defect detection, trained on historical codebases, helps identify insecure coding practices, risky resource usage, and dependency vulnerabilities. As a result, code entering production is likely to be more secure, reducing the attack surface.

Further, because the detection is automated and continuous, security becomes part of the development lifecycle (shift-left), reducing reliance on end-of-cycle audits, and enabling faster remediation. The result is not just fewer vulnerabilities, but earlier detection and fixed-before-release, which is critical in CI/CD environments with rapid, frequent deployments.

Improved Runtime Security and Rapid Detection of Intrusions

The runtime intrusion and anomaly detection module adds a dynamic layer of defense. By profiling normal behavior and continuously monitoring runtime telemetry, the system can detect novel attacks—including zero-day exploits, configuration drift, unauthorized access, or data exfiltration attempts—that static code analysis could not foresee. The result is a more robust, layered security posture capable of catching runtime misuses, malicious inputs, or unexpected network behavior as soon as they occur.

Automated alerts, isolation, or rollback mechanisms mean that potential breaches can be mitigated in real time, minimizing damage and downtime. This real-time responsiveness is especially valuable for critical infrastructure systems, where downtime or data breaches can have cascading, severe consequences.

Optimized Resources for Security Updates and Resilience Through Demand Forecasting

Incorporation of demand forecasting ensures that infrastructure resources are optimally allocated when deploying updates, patches, or security fixes. By predicting surges in resource demand—due to security patches, increased load during incident response, or scaling for mitigation efforts—the system can pre-provision necessary capacity. This reduces the risk of resource exhaustion, performance degradation, or deployment failures during critical security operations.

The result is improved resilience: security updates do not cause service disruptions; patches can be rolled out smoothly even under load; and the system maintains availability and performance even during security events. For organizations operating at scale—such as retail chains, supply networks, or IoT-enabled services—this capability is essential to maintain business continuity while ensuring robust security.

Alignment with NIST Framework and Compliance Readiness

By mapping each module to the core functions of the NIST Cybersecurity Framework (Identify, Protect, Detect, Respond, Recover), the proposed approach ensures that organizations not only implement state-of-the-art security mechanisms but also adhere to recognized governance standards (NIST, 2018). This alignment supports compliance, risk management, audit readiness, and accountability—and provides a structured, disciplined approach to cybersecurity in dynamic, AI-augmented environments.

Discussion

The proposed integrated framework seeks to bridge the divide between static cybersecurity standards and dynamic AI-driven security mechanisms within CI/CD-powered infrastructure systems. In doing so, it offers a comprehensive, multi-layered, automated approach. However, while theoretically robust, there are important limitations, challenges, and ethical considerations that warrant deep discussion, along with suggestions for future research.

Practical and Technical Challenges

First, implementing such a comprehensive, AI-driven security framework requires significant organizational commitment, technical expertise, and infrastructural investment. Many organizations may lack in-house AI expertise, or may lack resources to develop graph neural network models, build runtime telemetry systems, and deploy demand forecasting modules. The upfront cost and complexity may be prohibitive—particularly for small or mid-sized enterprises.

Second, AI-driven tools—including static code analyzers and anomaly detectors—are only as good as their training data and their integration. For code vulnerability detection, high-quality labeled datasets of vulnerable vs non-vulnerable code are essential. While research like Suneja et al. (2020) demonstrates viability of code-as-graph approaches, real-world codebases—especially in enterprise systems—are

diverse, include proprietary modules, external dependencies, third-party libraries, and legacy code. Training models that perform well across such heterogeneity is a significant challenge.

Moreover, false positives and false negatives are a persistent issue. Overly aggressive detection may generate many false positives, burdening developers with irrelevant alerts, reducing trust, and leading to alert fatigue. Conversely, false negatives can leave serious vulnerabilities undetected. Balancing precision and recall in AI detection models remains an open research challenge.

Third, runtime anomaly detection and intrusion detection in complex CI/CD-powered systems may generate enormous volumes of telemetry data. Collecting, storing, processing, and analyzing this data in real time requires scalable infrastructure, efficient data pipelines, and perhaps distributed AI models. For resource-constrained organizations, or those with limited bandwidth, this may be impractical.

Fourth, integrating demand forecasting for resource allocation introduces complexity in infrastructure management. Forecasts may be inaccurate, leading to over-provisioning (wasting resources), or under-provisioning (leading to failures during patch deployments). It also requires that infrastructure be elastic—perhaps using cloud resources—and that deployment and scaling mechanisms are robust.

Ethical and Governance Considerations

Beyond technical challenges, embedding AI-driven security systems raises ethical and governance considerations. Continuous code analysis and runtime monitoring might be viewed by developers or users as intrusive. Particularly in environments with human operators, extensive telemetry may risk privacy or raise concerns about surveillance. Organizations must ensure transparent policies, data minimization, access controls, and possibly anonymization to avoid misuse of monitoring data.

The use of AI in security also raises questions of accountability. If an AI-based analyzer fails to detect a vulnerability, or incorrectly flags benign code, responsibility remains with human operators. It is essential to maintain human-in-the-loop processes, comprehensive documentation, and review protocols to ensure accountability, reproducibility, and auditability.

Moreover, reliance on AI may create a false sense of security—a phenomenon known as automation complacency. Organizations might neglect traditional security practices, code reviews, or manual audits, assuming AI will catch all issues. This is dangerous; AI should augment—not replace—human judgment and conventional security hygiene.

Future Research Directions

Given the limitations, further research is required to operationalize and refine the proposed framework. Key directions include:

1. Dataset Creation and Standardization: Large-scale, labeled datasets of code vulnerabilities across diverse languages, frameworks, and deployment contexts are urgently needed. Shared datasets will accelerate development of robust models and enable benchmarking.

2. Model Robustness and Generalization: Investigate transfer learning, cross-language models, and domain adaptation techniques that allow vulnerability detectors to generalize across heterogeneous codebases—including legacy systems, third-party modules, and microservices.

3. Runtime Monitoring and Real-Time Analytics: Develop scalable, efficient data pipelines and distributed AI systems capable of real-time telemetry processing. Explore edge computing or lightweight agents for IoT and resource-constrained environments.

4. Human-in-the-Loop and Explainability: Enhance interpretability of AI detections—especially for code vulnerabilities and runtime anomalies—to enable human operators to understand why a given flag was raised. This helps in building trust and accountability.

5. Ethical Framework and Governance Models: Research governance models that balance security, privacy, transparency, and accountability in continuous monitoring environments. Institutional oversight mechanisms, data privacy protocols, and compliance with regulations (e.g., data protection) must be developed.

6. Integration with Business Operations: For sectors like retail, supply chain, and IoT, investigate how security modules interact with demand forecasting, inventory management, and operational logistics—to ensure security does not hinder, but supports operational efficiency and business goals (Malik et al., 2025).

7. Empirical Evaluation and Real-World Pilots: Field studies and pilot implementations in real CI/CD-powered critical infrastructure systems—including retail platforms, IoT networks, or energy systems—to measure actual effectiveness, performance overhead, and ROI of AI-centric security frameworks.

Conclusion

The accelerating pace of digital transformation has rendered traditional, manual, and reactive security practices inadequate. CI/CD pipelines, massive codebases, third-party dependencies, IoT proliferation, and AI-driven applications demand a paradigm shift in cybersecurity—from static audits to dynamic, continuous, AI-enhanced defense strategies. By integrating the conceptual rigor of the NIST Cybersecurity Framework with modern AI-based code analysis, runtime intrusion detection, and resource demand forecasting, the framework proposed in this paper offers a holistic, forward-looking approach to securing CI/CD-powered critical infrastructure systems.

While challenges remain—technical complexity, infrastructural demands, data requirements, governance and ethics—the potential benefits are profound: earlier vulnerability detection, real-time intrusion detection, resilient infrastructure, efficient resource allocation, and compliance readiness. Realizing this vision will require coordinated effort from researchers, developers, security professionals, and organizational leadership.

We call upon academic researchers, industry practitioners, and standard-setting bodies to collaborate: to build large-scale vulnerability datasets, develop robust AI models, pilot real-world integrations, and craft governance frameworks that balance security and privacy. Only then can we harness AI not just for speed and automation—but for resilience, safety, and sustainable security in the critical infrastructures of the future.

References

1. NIST - National Institute of Standards and Technology. (2018). Framework for improving critical infrastructure cybersecurity. Retrieved from https://doi.org/10.6028/NIST.CSWP.04162018
2. Kumar, S., Jain, A., Rani, S., Ghai, D., Achampeta, S., & Raja, P. (2021, December). Enhanced SBIR based Re-Ranking and Relevance Feedback. In 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART) (pp. 7-12). IEEE.
3. Malik, G., Rahul Brahmbhatt, & Prashasti. (2025). AI-Driven Security and Inventory Optimization: Automating Vulnerability Management and Demand Forecasting in CI/CD-Powered Retail Systems. International Journal of Computational and Experimental Science and Engineering, 11(3). https://doi.org/10.22399/ijcesen.3855
4. Jain, A., Singh, J., Kumar, S., Florin-Emilian, Ț., Traian Candin, M., & Chithaluru, P. (2022). Improved recurrent neural network schema for validating digital signatures in VANET. Mathematics, 10(20), 3895.
5. Kumar, S., Haq, M. A., Jain, A., Jason, C. A., Moparthi, N. R., Mittal, N., & Alzamil, Z. S. (2023). Multilayer Neural Network Based Speech Emotion Recognition for Smart Assistance. Computers, Materials & Continua, 75(1).
6. Misra, N. R., Kumar, S., & Jain, A. (2021, February). A review on E-waste: Fostering the need for green electronics. In 2021 international conference on computing, communication, and intelligent systems (ICCCIS) (pp. 1032-1036). IEEE.
7. Kumar, S., Shailu, A., Jain, A., & Moparthi, N. R. (2022). Enhanced method of object tracing using extended Kalman filter via binary search algorithm. Journal of Information Technology Management, 14(Special Issue: Security and Resource Management challenges for Internet of Things), 180-199.
8. Harshitha, G., Kumar, S., Rani, S., & Jain, A. (2021, November). Cotton disease detection based on deep learning techniques. In 4th Smart Cities Symposium (SCS 2021) (Vol. 2021, pp. 496-501). IET.
9. Jain, A., Dwivedi, R., Kumar, A., & Sharma, S. (2017). Scalable design and synthesis of 3D mesh network on chip. In Proceeding of International Conference on Intelligent Communication, Control and Devices: ICICCD 2016 (pp. 661-666). Springer Singapore.
10. Chen, T., et al. (2020). DeepCode: A Deep Learning Approach to Code Vulnerability Detection. IEEE Transactions on Software Engineering.
11. Ravindar Reddy Gopireddy. (2024). Securing the Future: The Convergence of Cybersecurity, AI, and IoT in a World Dominated by Intelligent Machines. European Journal of Advances in Engineering and Technology, 11(8), 91–95. https://doi.org/10.5281/zenodo.13753300
12. Szabó, Z., & Bilicki, V. (2023). A New Approach to Web Application Security: Utilizing GPT Language Models for Source Code Inspection. Future Internet. https://doi.org/10.3390/fi15100326
13. Ravindar Reddy Gopireddy. (2024). Securing AI Systems: Protecting Against Adversarial Attacks and Data Poisoning. Journal of Scientific and Engineering Research, 11(5), 276–281. https://doi.org/10.5281/zenodo.13253611
14. S, P., B, C., & Raju, L. (2022). Developer's Roadmap to Design Software Vulnerability Detection Model

Using Different AI Approaches. IEEE Access, 10, 75637-75656. https://doi.org/10.1109/access.2022.3191115

15. Ravindar Reddy Gopireddy. (2024). Ensuring Human-Centric AI: Ethical and Technical Safeguards for Collaborative Intelligence. European Journal of Advances in Engineering and Technology, 11(3), 125–130. https://doi.org/10.5281/zenodo.13253024
16. Borg, M., & Borg, M. (2023). Pipeline Infrastructure Required to Meet the Requirements on AI. IEEE Software, 40, 18-22. https://doi.org/10.1109/MS.2022.3211687
17. Gopireddy, R. R. (2018). MACHINE LEARNING FOR INTRUSION DETECTION SYSTEMS (IDS) AND FRAUD DETECTION IN FINANCIAL SERVICES [IJCEM Journal]. https://doi.org/10.5281/zenodo.13929200
18. Suneja, S., Zheng, Y., Zhuang, Y., Laredo, J., & Morari, A. (2020). Learning to map source code to software vulnerability using code-as-a-graph. ArXiv, abs/2006.08614