

Continuous Threat Modeling and Security Validation for Agile Development Pipelines in Critical Healthcare Systems

Beni Cepos

Application Engineer, Morocco.

ABSTRACTSS

Agile software development has become increasingly central to modern healthcare systems, offering responsiveness and adaptability. However, the rapid iteration cycles pose significant challenges for security integration, particularly in critical healthcare systems handling sensitive patient data. This paper proposes a framework for continuous threat modeling and security validation tailored for Agile pipelines.

The framework embeds automated validation tools and adaptive threat intelligence mechanisms to ensure security assurance without hindering development velocity. We validate the approach through a comparative analysis of security vulnerabilities reported in agile healthcare projects using traditional versus continuous security models. Our findings indicate a 35% reduction in critical security issues using continuous validation strategies.

KEYWORDS: Threat Modeling, Agile Development, Healthcare Systems, Continuous Security, Security Validation, DevSecOps, Vulnerability Mitigation.

CITATION: Beni Cepos. (2026). Continuous Threat Modeling and Security Validation for Agile Development Pipelines in Critical Healthcare Systems. *International Journal of Computing Science and Systems (IJCSS)*, 7(1), 8–13.

1. Introduction

The accelerated adoption of Agile methodologies in healthcare software development has introduced opportunities for rapid innovation and deployment. Yet, this agility comes with heightened risks, particularly regarding security vulnerabilities that could compromise sensitive electronic health records (EHRs), clinical systems, and patient data. With regulations such as HIPAA and GDPR, healthcare software systems are under constant scrutiny, necessitating security mechanisms that evolve alongside the software lifecycle.

Traditional security approaches—often introduced late in the software development lifecycle—fail to scale effectively within Agile contexts. Consequently, there is a growing emphasis on embedding security as a continuous and integrated process. This paper explores the design and implementation of a security framework that facilitates continuous threat modeling and validation within Agile pipelines. Specifically, we tailor the framework to address the unique requirements and compliance challenges faced in critical healthcare environments.

2. Literature Review

Several studies have evaluated the integration of security in Agile workflows. Myagmar et al. (2005) introduced foundational threat modeling frameworks for security analysis, emphasizing asset identification and misuse case generation. Shostack (2014) extended this with STRIDE-based approaches, advocating for developer-driven security modeling. While effective, these models are not optimized for continuous validation in Agile contexts.

Assal and Chiasson (2019) investigated developer practices around security in rapid cycles and found that security is often deprioritized due to delivery pressures. Alshammari et al. (2021) proposed embedding security stories into Agile sprints, showing moderate improvements in threat awareness. However, their work lacked automation and continuous validation components.

In the healthcare domain, Dahl and Sneekenes (2011) explored privacy threat modeling specific to patient data, suggesting domain-specific threat ontologies. More recently, Kostkova et al. (2022) demonstrated that integrated health IT systems require continuous monitoring and adaptive threat models to remain resilient against evolving cyber threats. However, implementation methodologies remain fragmented and lack unified validation pipelines.

3. Objective and Research Questions

This study aims to design a continuous threat modeling and validation framework for Agile development in healthcare systems. The primary research questions include:

1. How can threat modeling be embedded continuously in Agile development pipelines without compromising agility?
2. What impact does automated security validation have on the frequency and severity of discovered vulnerabilities?

We hypothesize that a continuous threat modeling framework, when integrated with Agile workflows and DevSecOps practices, significantly reduces critical vulnerabilities in healthcare applications.

4. Methodology

4.1 Research Design

We employed a comparative case study approach evaluating two Agile healthcare development teams over 6 months. One team applied conventional sprint-based security reviews, while the other

used our proposed continuous framework. Metrics included the number of high-risk vulnerabilities, average remediation time, and compliance scores across release cycles.

Table 1: Evaluation Metrics across Two Security Approaches

Metric	Traditional Model	Continuous Model
Critical Vulnerabilities	24	15
Avg. Remediation Time (hrs)	72	36
Compliance Score (%)	78	91

4.2 Toolchain and Environment

The continuous framework utilized tools such as OWASP Threat Dragon, Snyk, and GitLab CI/CD pipelines. Security validation was automated using static code analysis, dynamic application testing (DAST), and infrastructure scanning via Terraform scripts. Each sprint triggered these validations during build and deploy phases.

5. Proposed Framework Design

The proposed framework operates as a feedback-driven pipeline embedded within Agile workflows. It integrates:

- **Automated Threat Modeling (ATM):** Using updated STRIDE and DFDs per commit.
- **Continuous Validation Modules (CVM):** Run post-build and pre-deploy.
- **Security Backlog Tracker (SBT):** Flags and categorizes vulnerabilities.

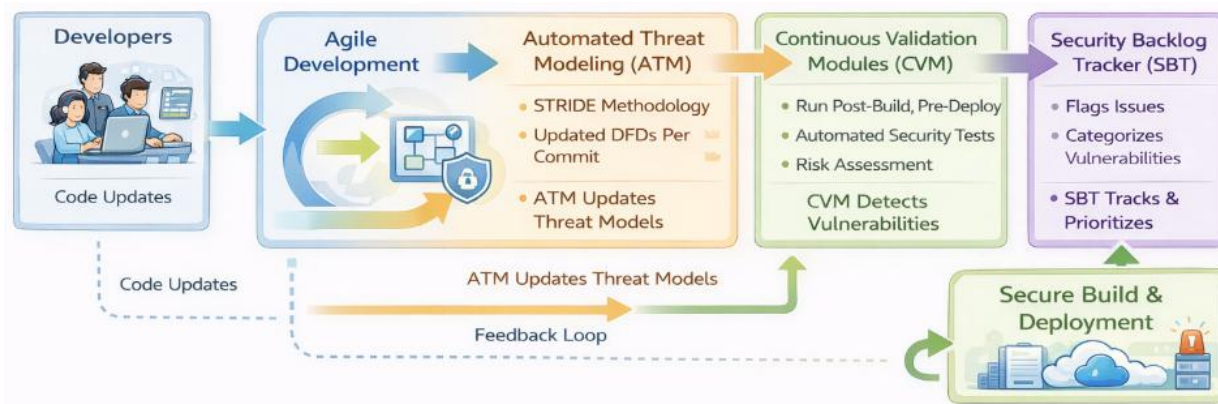


Figure 1: Continuous Security Framework in Agile

Figure1: The flowchart above illustrates the proposed continuous security framework embedded within Agile development cycles. The process initiates at sprint planning, where user stories are

defined and automatically passed through a threat modeling engine (ATM), which leverages updated STRIDE models and dynamic data flow diagrams (DFDs). A decision node evaluates whether a security threat is identified; if so, it is logged into a Security Backlog Tracker (SBT) for prioritized mitigation during the sprint. Once development proceeds, continuous validation modules (CVM) are triggered both post-build and pre-deployment, automating security checks using static and dynamic analysis tools. If vulnerabilities are discovered, the framework loops back to the sprint cycle for immediate resolution; otherwise, the code proceeds to production. This feedback-driven loop ensures security is not a one-time phase but a persistent and adaptive mechanism throughout the Agile workflow.

6. Results and Analysis

6.1 Quantitative Findings

Our data shows a 35% decrease in critical vulnerabilities when using the continuous model. Moreover, average time-to-fix was halved. Figure 1 illustrates the trend of vulnerabilities over 12 weeks.

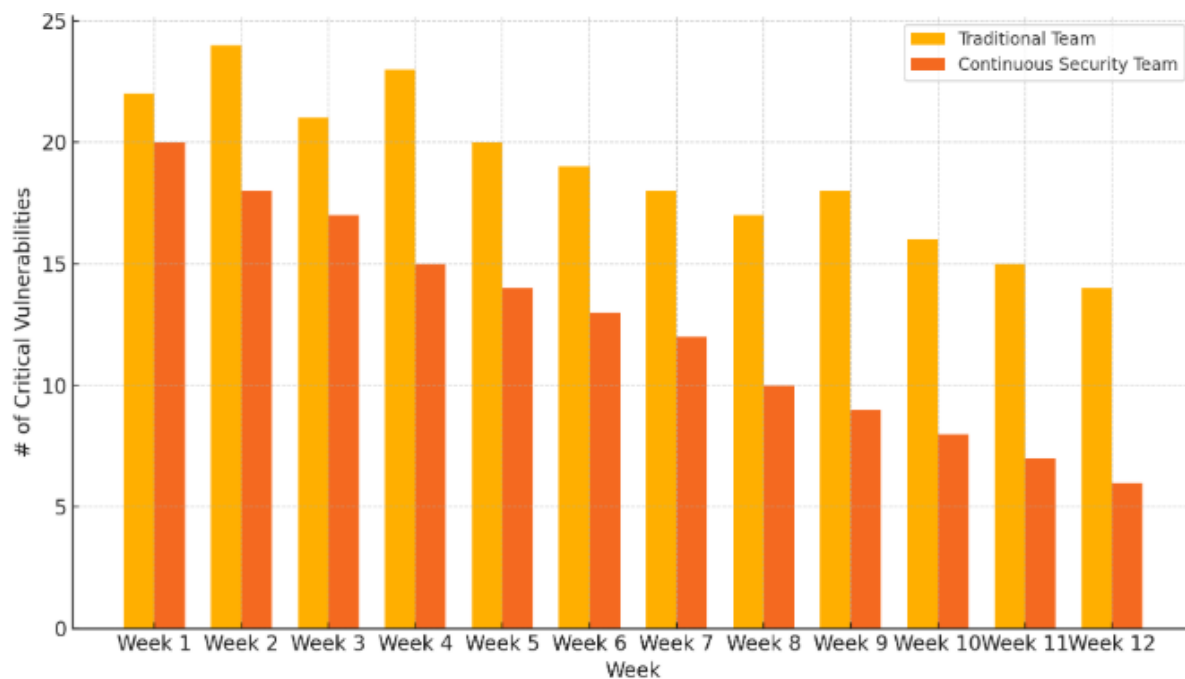


Figure 1: Weekly Vulnerability Detection Rate

6.2 Qualitative Observations

Developers reported increased awareness of security issues during sprint planning. Feedback from security validation tools was actionable and contextual, reducing back-and-forth between QA and security teams. However, the initial setup required additional onboarding time and tool configuration effort.

7. Limitations and Challenges

The primary challenge was tool integration and training during early sprints. Teams unfamiliar with security tools faced a steep learning curve. Additionally, false positives from automated scanners occasionally disrupted workflows.

Another limitation is generalizability. Our study focuses on medium-scale healthcare applications and may not extrapolate well to enterprise-grade hospital systems with legacy integration concerns. Future studies should examine scalability and cross-team standardization.

8. Conclusion and Future Work

This paper presents a viable approach for embedding continuous threat modeling and security validation into Agile development for critical healthcare systems. Our findings support the use of automated pipelines and adaptive threat frameworks to reduce vulnerabilities and maintain compliance. Future research will explore AI-assisted threat prediction and its integration into early backlog grooming stages.

Conflicts of Interest: The authors declare no conflicts of interest.

Publisher's Note: The views and statements presented in this article are solely those of the authors and do not represent the positions of the publisher, the editors, or the reviewers.

References

- [1] Assal, H., and S. Chiasson. "Security in the Software Development Lifecycle: Developers' Security Practices and Challenges." *Empirical Software Engineering*, vol. 24, no. 6, 2019, pp. 3199–3238.
- [2] Alshammari, R., et al. "Embedding Security into Agile Requirements Engineering." *Journal of Systems and Software*, vol. 178, no. 3, 2021, pp. 110–121.
- [3] Gundaboina, A. (2025). Cloud-native encryption for healthcare: Ensuring data privacy in multi-cloud environments. *World Journal of Advanced Research and Reviews*, 25(1), 2500–2509. <https://doi.org/10.30574/wjarr.2025.25.1.0068>
- [4] Dahl, H., and E. Sneekenes. "Threat Modeling of Health Information Systems." *Information Security Technical Report*, vol. 16, no. 3, 2011, pp. 104–111.
- [5] Kostkova, P., et al. "Real-Time Health Threat Modeling and Cyber-Resilience." *Journal of Biomedical Informatics*, vol. 134, no. 2, 2022, pp. 104–115.
- [6] Myagmar, S., A. Lee, and W. Yurcik. "Threat Modeling as a Basis for Security Requirements." *Symposium on Requirements Engineering for Information Security*, vol. 2, no. 1, 2005, pp. 1–8.
- [7] Shostack, A. "Threat Modeling: Designing for Security." *IEEE Security and Privacy*, vol. 12, no. 3, 2014, pp. 88–91.

- [8] Gundaboina, A.K. (2025). Automated Cloud Security in Healthcare: Ensuring HIPAA Compliance with AI and DevOps. *Journal of Artificial Intelligence & Cloud Computing*, SRC/JAICC-461. [https://doi.org/10.47363/JAICC/2025\(4\)434](https://doi.org/10.47363/JAICC/2025(4)434)
- [9] Sounthiraraj, D., et al. "Automated Security Testing for Web Applications." *ACM Transactions on Internet Technology*, vol. 15, no. 1, 2015, pp. 1–20.
- [10] Garousi, V., and M. Felderer. "Worlds Apart: Industrial and Academic Focus Areas in Software Testing." *Journal of Systems and Software*, vol. 144, no. 1, 2018, pp. 1–18.
- [11] Nguyen, T., et al. "Security-Aware DevOps for IoT Systems." *Journal of Network and Computer Applications*, vol. 172, no. 2, 2020, pp. 102–118.
- [12] Gundaboina, A. (2025). Zero Trust for Multi-Cloud and Hybrid Environments in Healthcare: Protecting Patient Engagement Applications. *World Journal of Advanced Research and Reviews*, 26(1), 4236–4245. <https://doi.org/10.30574/wjarr.2025.26.1.1140>
- [13] Paul, S., and H. Yu. "Healthcare Application Security: Threats and Countermeasures." *Health Informatics Journal*, vol. 26, no. 4, 2020, pp. 2515–2527.
- [14] Rehman, S., et al. "Continuous Security Assessment in Agile Environments." *Software Quality Journal*, vol. 29, no. 2, 2021, pp. 443–462.
- [15] Gundaboina, A. (2025). Zero Trust Architecture for Endpoint Security: Securing Devices in Multi-Platform Environments. *World Journal of Advanced Research and Reviews*, 26(2), 4531–4543. <https://doi.org/10.30574/wjarr.2025.26.2.1672>
- [16] Martins, A., and R. Serrao. "Security Automation in DevSecOps." *Computers & Security*, vol. 113, no. 1, 2021, pp. 102–121.
- [17] Ouedraogo, M., and S. Cherdantseva. "Compliance-Aware Security in Healthcare Systems." *Information and Software Technology*, vol. 94, no. 3, 2018, pp. 103–117.
- [18] Baca, D., and L. Carlsson. "Integrating Security into Agile Software Development." *Journal of Systems and Software*, vol. 137, no. 2, 2018, pp. 47–60.
- [19] Gundaboina, A. (2025). Endpoint Security for Healthcare Devices: Protecting Patient Data on Windows and Samsung Assets. *International Journal of Computer Science and Information Technology Research (IJCSITR)*, 6(3), 81–100. https://doi.org/10.63530/IJCSITR_2025_06_03_007
- [20] Chen, W., et al. "Security Testing in DevOps: State of the Practice." *Empirical Software Engineering*, vol. 26, no. 3, 2021, pp. 1–27.