Assignment 2.1: Designing a Secure Software Development Program

Joshua A. Wortz

University of San Diego

CSOL-599

Dr Moore

07/06/2026

Assignment 2.1: Designing a Secure Software Development Program

High-Level Solutions Design: Secure Software Development Program

Introduction and Organizational Context (The 5Ws)

As Acuity Health transitions to a rapidly expanding, cloud-hosted urgent care model, it is imperative to align our software development practices with our strategic business objectives of delivering highly available, secure patient care.

- **Who:** Acuity Health's small IT and Security teams, led by the CISO, are responsible for protecting the data of our patients, clinicians, and external pharmacy partners. Ultimate accountability for the program rests with the CISO, while development leads serve as the operational execution arm, establishing a clear separation of duties between security oversight and code production.
- **What:** The establishment of an enterprise cloud-native Secure Software Development Program to safeguard our clinical applications, telemedicine platforms, online check-in portals, and digital prescription APIs. This program operationalizes security baselines across all proprietary applications, shifting security left into the early phases of the engineering process.
- **Where**: Across our multi-site urgent care network, utilizing a Microsoft Azure cloud-hosted infrastructure governed by a Zero Trust Architecture (ZTA). This includes secure development landing zones, staging environments, and production clusters hosted in Azure Kubernetes Service (AKS).
- **When:** Security is integrated continuously throughout the entire Software Development Life Cycle (SDLC) via automated CI/CD pipelines, allowing us to maintain agility without sacrificing security (Agarwal & Ranjan, 2026).

Automated quality gates trigger on every code commit, pull request, and deployment.

- **Why:** To prevent sophisticated cyber threats, ensure uninterrupted patient safety, and maintain strict regulatory compliance with the Health Insurance Portability and Accountability Act (HIPAA) and the Payment Card Industry Data Security Standard (PCI DSS) (U.S. Department of Health and Human Services [HHS], 2022). This prevents financial liabilities and safeguards patient trust.

Securing the Development Space and Processes

Given our disproportionately small IT staff, securing the development space relies heavily on automation and alignment with our enterprise Zero Trust Architecture (ZTA). ZTA dictates that no actor, system, or environment is inherently trusted, requiring continuous authentication and least-privilege access across all environments (Agarwal & Ranjan, 2026). To secure the development processes, the organization will implement ephemeral build environments and ensure that access to source code repositories is strictly governed by Azure Active Directory using Role-Based Access Control (RBAC) and Multi-Factor Authentication (MFA). Furthermore, our development environment will be logically segmented; API gateways will act as Policy Enforcement Points (PEPs) within a Demilitarized Zone (DMZ) to ensure development traffic from moving laterally into critical production Electronic Medical Record (EMR) systems.

To address specific integrations and necessary diversions from the standard secure enterprise architecture, the development space will utilize isolated Azure DevTest Labs. While standard corporate users are restricted from accessing external code repositories, developers will have proxy-restricted outbound access to vetted public repositories (e.g., NuGet, npm) via a

secured Azure Firewall. To mitigate software supply chain risks, this traffic is strictly channeled through an artifact repository manager that caches and dynamically scans external packages for vulnerabilities before allowing them into the ephemeral build environment.

Secure Software Development Framework (SSDF)

Acuity Health will utilize the NIST Secure Software Development Framework (SP 800-218 v1.1) (National Institute of Standards and Technology [NIST], 2022). This outcome-based framework provides a set of fundamental, sound, secure software development practices that integrate seamlessly into our DevSecOps SDLC. The SSDF consists of four core practice groups that we will implement:

- **Prepare the Organization (PO):** Ensure our people, processes, and technology are equipped to perform secure development. Acuity Health will institute annual developer training focused on the OWASP Top 10 and formalize a "Security Champions" program in which one developer serves as a liaison to the security team.
- **Protect the Software (PS):** Safeguarding all components of the software from tampering and unauthorized access. We will enforce cryptographic signing of build artifacts and generate a comprehensive Software Bill of Materials (SBOM) for every release to ensure code integrity.
- **Produce Well-Secured Software (PW):** Designing software to minimize vulnerabilities from the onset. This will be achieved by requiring documented threat modeling sessions for all major architecture revisions and implementing standardized, pre-approved secure coding libraries for authentication and encryption.

- **Respond to Vulnerabilities (RV):** Identifying residual vulnerabilities in software releases. We will establish a formal Vulnerability Disclosure Program (VDP) and coordinate patch-response SLAs (e.g., critical bugs patched within 48 hours) to ensure rapid remediation.

Secure Design and Coding Principles

To ensure our telemedicine and API platforms are resilient, our development team will adopt the OWASP Application Security Verification Standard (ASVS) and the OWASP Developer Guide (OWASP Foundation, 2023). Our secure coding practices will explicitly target mitigation of the OWASP API Security Top 10, particularly API1: Broken Object Level Authorization (BOLA), the leading vulnerability that enables attackers to manipulate object identifiers to access unauthorized patient data. Key coding principles will include:

- **Input Validation:** Enforcing strict data validation schemas (parameterized inputs, allow-listing, and regular expression validation) for all online check-ins to eliminate injection attacks.
- **Data Protection & Cryptography:** Ensuring all electronic Protected Health Information (ePHI) is encrypted at rest using AES-256 and in transit using TLS 1.3 with secure cipher suites to meet HIPAA and PCI DSS requirements (HHS, 2022). Database fields containing patient IDs will use cryptographic salts.
- **Access Control:** Implementing strong digital identity verification and avoiding weak API keys. To explicitly mitigate API1: Broken Object Level Authorization (BOLA), the system will replace sequential integer object identifiers with cryptographically secure Universally Unique Identifiers (UUIDs). Furthermore,

an access control context check will be enforced at the data layer to verify that the logged-in user has explicit permission to access the requested resource ID.

Threat Modeling

Threat modeling is the proactive process of using hypothetical scenarios and system diagrams to identify architectural vulnerabilities before code is written. For Acuity Health, we will utilize the STRIDE framework (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege). Threat modeling is vital because it allows us to visualize our clinical workflows, such as a patient's lab results moving from the EMR to a third-party application, using Data Flow Diagrams (DFDs) (Censinet, 2026). By mapping these trust boundaries, we can identify risks tied to the exposure of PHI (Information Disclosure) or interruptions to clinical systems (Denial of Service) early in the design phase. STRIDE is highly recommended by the Centers for Medicare & Medicaid Services (CMS) for healthcare environments because its categories map directly to HIPAA's core principles of confidentiality, integrity, and availability.

Threat modeling will be formalized as an immutable gate within our Agile sprint planning phase. Before any user story involving ePHI ingestion or third-party data exchange is approved for development, a mini-STRIDE assessment must be documented. The outputs of these modeling sessions, identified threats and corresponding mitigations, will be logged directly as security user stories in our development backlog, ensuring that threat mitigation is tracked with the same velocity as functional feature engineering.

Testing and Validation Processes

To combat the growing attack surface without overwhelming our small team, security testing will be heavily automated and integrated into our continuous CI/CD pipeline (Agarwal & Ranjan, 2026). Our testing and validation processes will include:

- **Static Application Security Testing (SAST):** Automated tools will scan source code during the build phase to detect known insecure patterns, such as hardcoded credentials or missing input validations, before the software is compiled.
- **Software Composition Analysis (SCA):** SCA tools automatically scan open-source libraries and third-party API dependencies for known vulnerabilities (CVEs), helping protect our software supply chain.
- **Dynamic Application Security Testing (DAST):** DAST will simulate an attacker's approach in test environments by executing the API under test and probing it for real-world runtime vulnerabilities, such as authentication bypasses or BOLA flaws.

To prevent alert fatigue within our small team while maintaining a strict security posture, these tools will enforce automated Quality Gates within the Azure DevOps CI/CD pipeline. The discovery of any Critical or High-severity vulnerability during SAST or SCA scans will automatically fail the build, preventing the code from merging into the main branch. For Medium and Low bugs, automated Jira tickets will be created, giving developers a 30-day window to remediate before triggering an automated deployment block.

Conducting a Mock Risk Assessment (ISO27001:2013)

To ensure a comprehensive evaluation of both operational and cyber risks, Acuity Health follows the risk assessment guidelines established by ISO/IEC 27001:2013 (Vasudevan, 2015). Executing a proper risk assessment requires following a structured methodology that includes establishing risk criteria, identifying threats and vulnerabilities to ePHI, identifying risk owners, and assessing the likelihood and impact of those events. By calculating overall risk levels and comparing them against acceptable organizational thresholds, the CISO can prioritize risks for treatment (e.g., terminate, transfer, or mitigate controls) and retain documentation to ensure consistent, valid results for future audits.

The 10-Step ISO 27001 Risk Assessment Methodology

To fulfill the standard's rigorous requirements, Acuity Health operationalizes its risk assessment utilizing the following ten iterative steps:

1. **Define the Risk Assessment Framework:** Establish the baseline criteria for evaluating risk, ensuring scales for likelihood and impact are consistent and aligned with organizational objectives.
2. **Identify Assets to Scope:** Catalog all information assets, repositories, cloud endpoints, and infrastructure components that process, transmit, or store ePHI or operational telemetry.
3. **Identify Threats:** Enumerate active, passive, internal, and external threats capable of compromising organizational security, ranging from malicious state actors to human error.

4. **Identify Vulnerabilities:** Discover structural, technical, or procedural security weaknesses within the application layer, CI/CD pipelines, or personnel configurations that could be exploited by threats.
5. **Analyze the Impact:** Assess the exact business, legal, financial, and clinical consequences if a specific threat successfully exploits an identified vulnerability.
6. **Determine Likelihood:** Evaluate the statistical probability that a threat event will occur based on historical metrics, industry trends, and current control efficacy.
7. **Calculate the Risk Level:** Quantify the total risk exposure by systematically combining the analyzed impact and likelihood values against the established matrix.
8. **Compare Risk Against Thresholds:** Evaluate the calculated risk scores against the organization's pre-defined risk appetite to determine which risks require immediate mitigation.
9. **Prioritize and Own:** Assign explicit accountability to designated Risk Owners who possess the operational authority to manage, fund, and oversee the specific risk vector.
10. **Document and Treat:** Establish a formal Risk Treatment Plan (RTP), choosing to mitigate, accept, transfer, or avoid each identified risk, and log all findings in the master Risk Register for audit validation.

References

Agarwal, M., & Ranjan, K. (2026). *Security-first approach to API pipeline development with zero-trust architecture.* arXiv preprint arXiv:2606.09062.

Censinet. (2026). *Healthcare-specific threat modeling frameworks.* https://censinet.com/capabilities/artificial-intelligence-governance

National Institute of Standards and Technology. (2022). *Secure software development framework (SSDF) version 1.1: Recommendations for mitigating the risk of software vulnerabilities (NIST Special Publication 800-218).* U.S. Department of Commerce.

OWASP Foundation. (2023). *OWASP top 10 API security risks – 2023* - Open Web Application Security Project.

Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero trust architecture (NIST Special Publication 800-207).* National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-207

U.S. Department of Health and Human Services. (2022). *Health information privacy: Summary of the HIPAA security rule.*

Vasudevan, V. (2015). *Application security in the ISO27001: 2013 environment.* IT Governance Ltd.