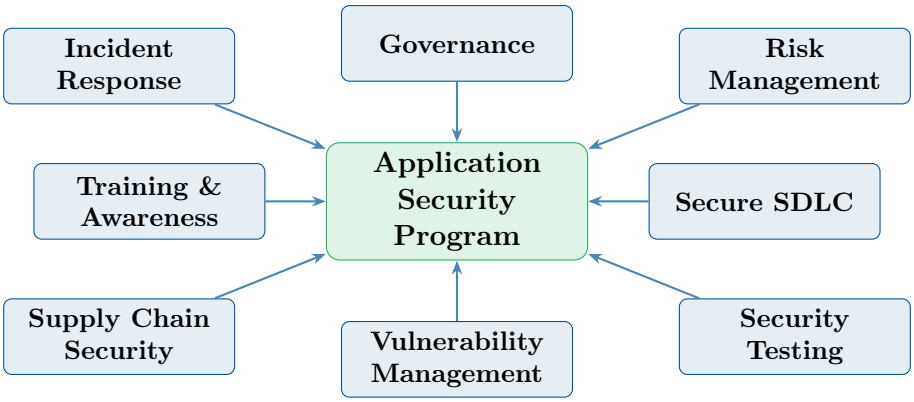
Application Security Program

Comprehensive Framework for Enterprise Security



Technology-Agnostic Framework

Applicable across platforms, languages, architectures, and deployment models

Version 1.0

December 3, 2025

Contents

1 Executive Summary **5**
1.1 Program Objectives . . . 5
1.2 Scope and Applicability . . . 6
1.3 Program Structure Overview . . . 6

2 Governance Framework **7**
2.1 Governance Principles . . . 7
2.2 Organizational Structure . . . 8
2.2.1 Executive Sponsorship . . . 8
2.2.2 Security Steering Committee . . . 9
2.2.3 Application Security Team Structure . . . 9
2.3 Roles and Responsibilities . . . 10
2.3.1 Development Team Responsibilities . . . 10
2.3.2 Security Champions . . . 11
2.4 Policy Framework . . . 11
2.4.1 Policy Hierarchy . . . 11
2.4.2 Core Security Policies . . . 12
2.4.3 Exception Management . . . 13

3 Risk Management **14**
3.1 Risk Management Framework . . . 14
3.2 Risk Identification . . . 14
3.3 Risk Assessment Methodology . . . 15
3.3.1 Likelihood Assessment . . . 15
3.3.2 Impact Assessment . . . 15
3.3.3 Risk Scoring . . . 16
3.4 Risk Appetite and Tolerance . . . 17
3.5 Risk Treatment . . . 18
3.6 Application Risk Classification . . . 19

4 Secure Software Development Lifecycle **20**
4.1 SSDLC Overview . . . 20
4.2 Requirements Phase . . . 20
4.2.1 Security Requirements Identification . . . 20
4.2.2 Security Requirements Categories . . . 21
4.2.3 Requirements Phase Deliverables . . . 21
4.3 Design Phase . . . 22
4.3.1 Security Architecture Principles . . . 22
4.3.2 Threat Modeling . . . 23
4.3.3 Security Design Review . . . 24
4.3.4 Design Phase Deliverables . . . 24
4.4 Development Phase . . . 25
4.4.1 Secure Coding Practices . . . 25
4.4.2 Code Review . . . 26
4.4.3 Static Application Security Testing (SAST) . . . 26
4.4.4 Software Composition Analysis (SCA) . . . 27

4.4.5 Secrets Management . . . 27
4.4.6 Development Phase Deliverables . . . 28
4.5 Testing Phase . . . 29
4.5.1 Dynamic Application Security Testing (DAST) . . . 29
4.5.2 Interactive Application Security Testing (IAST) . . . 29
4.5.3 Penetration Testing . . . 30
4.5.4 Security Test Case Development . . . 30
4.5.5 Testing Phase Deliverables . . . 30
4.6 Deployment Phase . . . 31
4.6.1 Security Configuration Management . . . 31
4.6.2 Deployment Security Checklist . . . 31
4.6.3 Security Gate Approval . . . 32
4.7 Operations and Maintenance Phase . . . 33
4.7.1 Continuous Security Monitoring . . . 33
4.7.2 Patch and Update Management . . . 33
4.7.3 Application Retirement . . . 33

5 Security Testing Program 34
5.1 Testing Program Objectives . . . 34
5.2 Testing Coverage Model . . . 34
5.3 Automated Security Testing . . . 35
5.3.1 CI/CD Integration . . . 35
5.3.2 Scan Scheduling and Triggers . . . 35
5.4 Manual Security Testing . . . 36
5.4.1 Penetration Testing Methodology . . . 36
5.4.2 Red Team Exercises . . . 36
5.4.3 Bug Bounty Program . . . 37
5.5 Vulnerability Severity Classification . . . 37

6 Vulnerability Management 38
6.1 Vulnerability Management Lifecycle . . . 38
6.2 Vulnerability Identification . . . 38
6.3 Vulnerability Assessment and Prioritization . . . 39
6.3.1 Assessment Factors . . . 39
6.3.2 Prioritization Model . . . 39
6.4 Remediation Requirements . . . 40
6.4.1 Service Level Agreements . . . 40
6.4.2 Remediation Options . . . 40
6.5 Verification and Closure . . . 40
6.6 Vulnerability Tracking and Reporting . . . 41
6.6.1 Tracking Requirements . . . 41
6.6.2 Reporting Cadence . . . 41

7 Third-Party and Supply Chain Security 42
7.1 Supply Chain Risk Landscape . . . 42
7.2 Vendor Security Assessment . . . 42
7.2.1 Assessment Requirements . . . 42
7.2.2 Assessment Process . . . 43

7.3 Open Source Security . . . 44
7.3.1 Open Source Governance . . . 44
7.3.2 Dependency Management . . . 44
7.4 Software Bill of Materials (SBOM) . . . 45
7.4.1 SBOM Requirements . . . 45
7.5 Build Pipeline Security . . . 45

8 Security Training and Awareness 46
8.1 Training Program Objectives . . . 46
8.2 Role-Based Training Requirements . . . 46
8.3 Developer Security Training . . . 47
8.3.1 Secure Coding Curriculum . . . 47
8.3.2 Training Delivery Methods . . . 47
8.4 Security Champion Program Training . . . 48
8.5 Training Effectiveness Measurement . . . 48

9 Application Security Incident Response 49
9.1 Incident Response Objectives . . . 49
9.2 Incident Classification . . . 49
9.3 Incident Response Process . . . 50
9.3.1 Detection . . . 50
9.3.2 Analysis . . . 50
9.3.3 Containment . . . 51
9.3.4 Eradication . . . 51
9.3.5 Recovery . . . 51
9.3.6 Post-Incident Review . . . 51
9.4 Communication Requirements . . . 52

10 Compliance and Audit 53
10.1 Compliance Framework . . . 53
10.1.1 Regulatory Requirements . . . 53
10.1.2 Industry Standards . . . 53
10.2 Control Documentation . . . 54
10.2.1 Control Library . . . 54
10.2.2 Evidence Management . . . 54
10.3 Audit Support . . . 55
10.3.1 Internal Audit . . . 55
10.3.2 External Audit . . . 55
10.4 Compliance Monitoring . . . 55

11 Metrics and Reporting 56
11.1 Metrics Framework . . . 56
11.1.1 Metrics Categories . . . 56
11.2 Key Performance Indicators . . . 57
11.2.1 Primary KPIs . . . 57
11.2.2 Key Risk Indicators . . . 57
11.3 Reporting Structure . . . 58
11.3.1 Operational Reporting . . . 58

11.3.2 Management Reporting . . . 58
11.3.3 Executive Reporting . . . 58
11.4 Continuous Improvement . . . 59

12 Program Maturity **60**
12.1 Maturity Model . . . 60
12.2 Maturity Domains . . . 61
12.3 Maturity Assessment Process . . . 61
12.4 Maturity Improvement Roadmap . . . 62

13 Conclusion **63**

A Appendix A: Security Requirements Checklist **64**

B Appendix B: Threat Modeling Template **65**

C Appendix C: Vulnerability Report Template **66**

D Appendix D: Security Tool Categories **67**

E Appendix E: Reference Standards and Frameworks **68**

1 Executive Summary

This Application Security Program establishes the comprehensive framework, governance structure, processes, and controls necessary to protect the organization's software applications throughout their entire lifecycle. The program takes a technology-agnostic approach, providing principles and practices applicable across any platform, programming language, architecture, or deployment model.

Program Mission Statement

To systematically reduce application security risk across the enterprise by embedding security into every phase of the software development lifecycle, fostering a security-conscious culture, and establishing measurable controls that protect organizational assets, customer data, and business operations from application-layer threats.

1.1 Program Objectives

The Application Security Program pursues the following strategic objectives:

1. **Risk Reduction:** Systematically identify, assess, and mitigate application security risks before they can be exploited, reducing the organization's overall threat exposure.

2. **Secure Development:** Integrate security activities into all phases of the software development lifecycle, ensuring that security is a fundamental consideration rather than an afterthought.

3. **Vulnerability Management:** Establish efficient processes for discovering, prioritizing, remediating, and verifying the closure of security vulnerabilities across the application portfolio.

4. **Compliance Assurance:** Meet regulatory, contractual, and industry security requirements through documented controls, regular assessments, and audit-ready evidence.

5. **Security Culture:** Build organization-wide security awareness and capability through targeted training, clear accountability, and positive reinforcement of secure practices.

6. **Continuous Improvement:** Evolve program maturity through regular assessment, metrics-driven decision making, and adoption of emerging best practices.

1.2 Scope and Applicability

This program applies to all software applications developed, acquired, deployed, or maintained by the organization, including:

- Internally developed applications regardless of technology stack
- Commercial off-the-shelf (COTS) software with custom configurations
- Software-as-a-Service (SaaS) applications integrated into business processes
- Mobile applications across all platforms
- Application programming interfaces (APIs) both internal and external
- Microservices and containerized workloads
- Serverless functions and cloud-native applications
- Legacy systems and mainframe applications
- Third-party components and open-source libraries
- Infrastructure-as-code and configuration scripts

1.3 Program Structure Overview

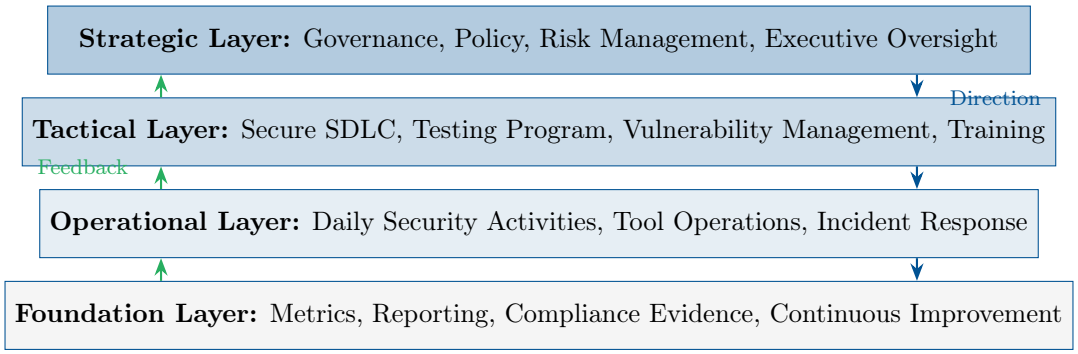


Figure 1: Application Security Program Layered Structure

2 Governance Framework

Effective governance provides the foundation for a successful Application Security Program. This section establishes the organizational structures, roles, responsibilities, and decision-making processes that ensure security activities align with business objectives and receive appropriate oversight.

2.1 Governance Principles

The Application Security Program governance adheres to the following principles:

Core Governance Principles

1. **Accountability:** Every security control, process, and decision has a clearly defined owner responsible for its effectiveness.
2. **Transparency:** Security posture, risks, and program performance are visible to appropriate stakeholders through regular reporting.
3. **Risk-Based Prioritization:** Resources and attention are allocated based on assessed risk to business objectives rather than compliance checklists alone.
4. **Business Alignment:** Security requirements and activities support rather than obstruct legitimate business operations.
5. **Continuous Oversight:** Regular review cycles ensure the program remains effective and adapts to changing threats and business needs.
6. **Segregation of Duties:** Critical security functions maintain appropriate separation to prevent conflicts of interest and single points of failure.

2.2 Organizational Structure

2.2.1 Executive Sponsorship

Executive sponsorship is essential for program success. The executive sponsor, typically a C-level officer such as the Chief Information Security Officer (CISO), Chief Technology Officer (CTO), or Chief Information Officer (CIO), provides strategic direction, secures budget allocation, removes organizational obstacles, and ensures security priorities receive appropriate attention at the executive level.

Responsibility	Description
Strategic Direction	Define the overall vision and strategic objectives for application security aligned with enterprise risk appetite
Resource Allocation	Secure and allocate budget, personnel, and tooling necessary for program execution
Organizational Authority	Provide the authority necessary to enforce security requirements across business units
Executive Communication	Report program status, risks, and achievements to the board and executive leadership
Conflict Resolution	Resolve disputes between security requirements and business priorities
Culture Leadership	Champion security as an organizational value through visible support and communication

Table 1: Executive Sponsor Responsibilities

2.2.2 Security Steering Committee

The Security Steering Committee provides cross-functional oversight and governance for the Application Security Program. This body includes representatives from key stakeholder groups and meets regularly to review program performance, address strategic issues, and align security activities with business objectives.

Steering Committee Composition

- Executive Sponsor (Chair)
- Application Security Program Manager
- Engineering/Development Leadership
- IT Operations Leadership
- Enterprise Architecture Representative
- Legal/Compliance Representative
- Business Unit Representatives (rotating)
- Risk Management Representative
- Internal Audit Representative (observer)

The Steering Committee convenes monthly for operational reviews and quarterly for strategic planning sessions. Committee responsibilities include approving program strategy and roadmap, reviewing risk acceptance decisions above defined thresholds, resolving cross-functional conflicts, evaluating program metrics and maturity assessments, and approving significant policy changes.

2.2.3 Application Security Team Structure

The Application Security team executes the day-to-day activities of the program. Team size and structure scale with organizational size, application portfolio complexity, and program maturity. The following model represents a mature team structure that organizations should evolve toward over time.

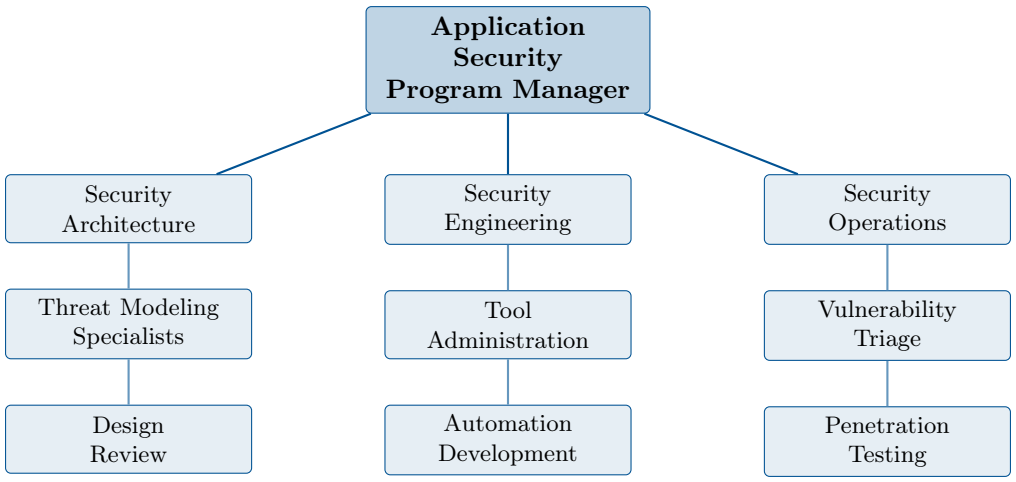


Figure 2: Application Security Team Organization

2.3 Roles and Responsibilities

Clear definition of roles and responsibilities ensures accountability and prevents gaps in security coverage. The following RACI matrix defines responsibility assignments for key program activities.

Activity	Exec	AppSec	Dev	Ops	Legal	Audit
Security Policy Approval	A	R	C	C	C	I
Risk Assessment	A	R	C	C	I	I
Threat Modeling	I	A/R	R	C	—	—
Secure Code Review	I	A	R	—	—	—
Security Testing	I	A/R	C	C	—	I
Vulnerability Remediation	I	C	A/R	R	—	I
Security Training	A	R	R	R	—	—
Incident Response	A	R	R	R	C	I
Compliance Reporting	A	R	C	C	R	R
Tool Selection	A	R	C	C	C	—

Table 2: RACI Matrix for Application Security Activities (R=Responsible, A=Accountable, C=Consulted, I=Informed)

2.3.1 Development Team Responsibilities

Development teams bear primary responsibility for building secure software. This includes writing secure code, remediating identified vulnerabilities within SLA timeframes, participating in security training, incorporating security requirements into user stories and acceptance criteria, and escalating potential security concerns to the Application Security team.

2.3.2 Security Champions

Security Champions are development team members who serve as the primary security liaison between their team and the Application Security function. This distributed model extends security expertise across the organization and embeds security considerations into daily development activities.

Security Champion Program Requirements

Selection Criteria:

- Demonstrated interest in security topics
- Respected technical contributor within their team
- Effective communicator across technical and non-technical audiences
- Sufficient tenure to understand team processes and codebase

Champion Responsibilities:

- Participate in advanced security training (minimum 20 hours annually)
- Attend monthly Security Champion meetings
- Serve as first point of contact for security questions within the team
- Promote security awareness and best practices
- Assist with vulnerability triage and remediation prioritization
- Participate in threat modeling sessions for team projects
- Provide feedback on security tools and processes

Program Support:

- Dedicated time allocation (10-20% of capacity)
- Access to advanced training and certifications
- Recognition in performance evaluations
- Direct communication channel to Application Security leadership

2.4 Policy Framework

2.4.1 Policy Hierarchy

The Application Security Program operates within a hierarchical policy framework that ensures consistency while allowing appropriate flexibility at operational levels.

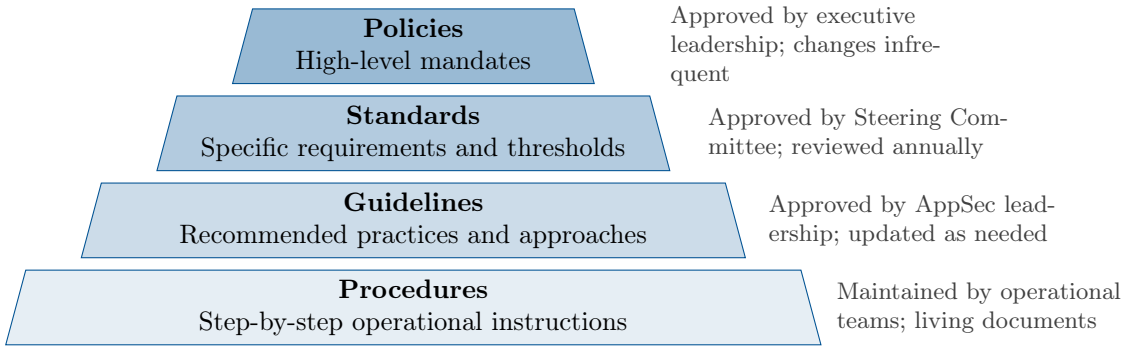


Figure 3: Policy Document Hierarchy

2.4.2 Core Security Policies

The following policies form the foundation of the Application Security Program:

Policy	Scope and Purpose
Secure Development Policy	Establishes requirements for integrating security into all phases of software development, including mandatory security activities, training requirements, and quality gates
Vulnerability Management Policy	Defines requirements for identifying, assessing, prioritizing, remediating, and reporting security vulnerabilities across the application portfolio
Security Testing Policy	Specifies mandatory security testing activities, frequency requirements, scope coverage, and finding management procedures
Third-Party Security Policy	Establishes requirements for assessing and managing security risks associated with vendor software, open-source components, and external services
Security Architecture Policy	Defines security design principles, approved patterns, reference architectures, and design review requirements for applications
Data Protection Policy	Specifies requirements for protecting sensitive data within applications, including classification, encryption, access controls, and retention
Authentication and Authorization Policy	Establishes requirements for identity verification, access control implementation, session management, and privilege management
Security Incident Response Policy	Defines procedures for detecting, analyzing, containing, and recovering from application security incidents
Security Exception Policy	Establishes the process for requesting, reviewing, approving, and tracking deviations from security requirements
Security Metrics and Reporting Policy	Defines requirements for measuring, reporting, and acting upon application security metrics

2.4.3 Exception Management

Security exceptions allow controlled deviation from policy requirements when business justification exists and compensating controls adequately mitigate residual risk. The exception process ensures that deviations are documented, time-limited, monitored, and approved at appropriate authority levels.

Risk Level	Approval Authority	Maximum Duration	Review Frequency
Critical	Executive Sponsor	30 days	Weekly
High	Steering Committee	90 days	Bi-weekly
Medium	AppSec Manager	180 days	Monthly
Low	AppSec Team Lead	365 days	Quarterly

Table 4: Exception Approval Authority Matrix

Exception requests must include business justification for the deviation, risk assessment of the exception, proposed compensating controls, remediation plan with target date, and business owner acceptance of residual risk. All exceptions are tracked in a central register and reported to the Steering Committee.

3 Risk Management

Application security risk management provides the framework for identifying, assessing, prioritizing, and treating security risks across the application portfolio. This risk-based approach ensures that limited security resources focus on the highest-impact activities.

3.1 Risk Management Framework

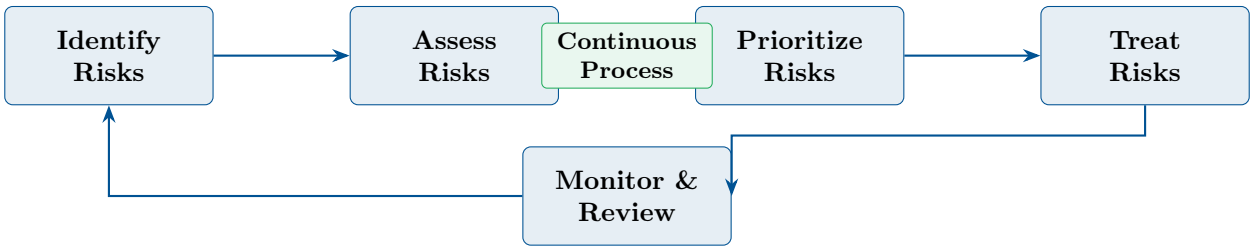


Figure 4: Risk Management Lifecycle

3.2 Risk Identification

Risk identification systematically discovers potential threats to application security. Sources for risk identification include:

- Threat intelligence feeds and industry reports
- Vulnerability scan and penetration test results
- Security incident post-mortems
- Architecture and design reviews
- Threat modeling exercises
- Audit findings and compliance assessments
- Third-party security assessments
- Bug bounty program submissions
- Developer and operations team input
- Changes in regulatory or contractual requirements

3.3 Risk Assessment Methodology

Risk assessment evaluates identified risks based on likelihood of occurrence and potential impact to the organization. This program uses a semi-quantitative approach that balances rigor with practicality.

3.3.1 Likelihood Assessment

Likelihood reflects the probability that a vulnerability will be exploited or a threat will materialize, considering threat actor capability and motivation, exposure and accessibility of the vulnerable component, existing controls and their effectiveness, and historical incident data.

Level	Score	Description
Very High	5	Exploitation expected; active exploitation observed in the wild; trivial to exploit
High	4	Exploitation likely; exploit code available; minimal skill required
Medium	3	Exploitation possible; requires moderate skill or specific conditions
Low	2	Exploitation unlikely; requires significant skill, access, or rare conditions
Very Low	1	Exploitation improbable; theoretical only or requires extraordinary circumstances

Table 5: Likelihood Rating Scale

3.3.2 Impact Assessment

Impact assessment considers multiple dimensions of potential harm:

Impact Category	Considerations
Financial	Direct financial loss, regulatory fines, remediation costs, legal liability
Operational	Business process disruption, system availability, recovery time
Reputational	Customer trust, brand damage, media exposure, market position
Legal/Regulatory	Compliance violations, contractual breaches, litigation exposure
Safety	Physical safety of individuals, environmental impact
Data	Confidentiality breach scope, data sensitivity, affected individuals

Table 6: Impact Assessment Categories

Level	Score	Description
Critical	5	Existential threat to organization; massive breach; regulatory shutdown
High	4	Major financial loss; significant breach; substantial regulatory action
Medium	3	Moderate financial impact; limited breach; regulatory inquiry
Low	2	Minor financial impact; minimal data exposure; internal concern
Minimal	1	Negligible impact; no data exposure; no regulatory implications

Table 7: Impact Rating Scale

3.3.3 Risk Scoring

Risk score is calculated as the product of likelihood and impact scores, producing a value between 1 and 25:

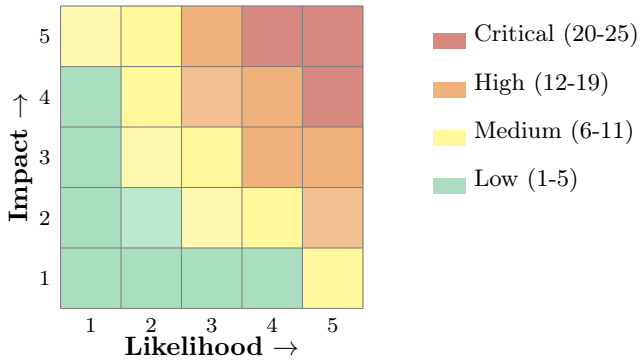


Figure 5: Risk Scoring Matrix

3.4 Risk Appetite and Tolerance

Risk appetite defines the aggregate level of risk the organization is willing to accept in pursuit of its objectives. Risk tolerance defines acceptable variation around specific risk categories or activities.

Risk Appetite Statement

The organization maintains a **conservative risk appetite** for application security, recognizing that application-layer vulnerabilities represent a primary attack vector and that security incidents can cause significant financial, reputational, and regulatory harm. The organization accepts that some residual risk is unavoidable but commits to maintaining risk within defined tolerance levels through systematic identification, assessment, and mitigation of application security risks.

Risk Category	Tolerance	Rationale
Critical Vulnerabilities	Zero	No critical vulnerabilities accepted in production systems
High Vulnerabilities	Very Low	Limited tolerance with strict SLA and executive visibility
Data Breach Risk	Very Low	Strict controls required for all sensitive data processing
Compliance Risk	Low	Regulatory violations carry significant penalties
Third-Party Risk	Moderate	Accept managed risk for business-critical vendors
Emerging Technology Risk	Moderate	Accept higher uncertainty for innovation initiatives

Table 8: Risk Tolerance by Category

3.5 Risk Treatment

Risk treatment selects and implements controls to modify risk. Four treatment options exist:

Risk Treatment Options

Avoid: Eliminate the risk by discontinuing the activity or removing the vulnerable component. Appropriate when risk exceeds appetite and no acceptable mitigation exists.
Mitigate: Implement controls to reduce likelihood or impact. The primary treatment option for most application security risks.
Transfer: Share risk with third parties through insurance, contracts, or outsourcing. Does not eliminate risk but redistributes financial consequences.
Accept: Acknowledge and monitor risk without additional controls. Requires documented acceptance by appropriate authority based on risk level.

3.6 Application Risk Classification

Applications are classified based on their risk profile to ensure appropriate security treatment. Classification considers data sensitivity, business criticality, exposure, and regulatory requirements.

Tier	Criteria	Security Requirements
Tier 1 (Critical)	Processes highly sensitive data; revenue-critical; customer-facing; regulatory scope	Full security testing suite; annual penetration test; threat modeling; security architecture review; continuous monitoring
Tier 2 (High)	Processes sensitive data; supports critical business functions; internal enterprise applications	SAST/DAST; periodic penetration testing; security design review; vulnerability scanning
Tier 3 (Medium)	Processes internal data; supports departmental functions; limited exposure	Automated security scanning; standard secure development practices; risk-based testing
Tier 4 (Low)	Minimal data processing; limited business impact; no external exposure	Basic security hygiene; standard development practices; periodic review

Table 9: Application Risk Classification Tiers

4 Secure Software Development Lifecycle

The Secure Software Development Lifecycle (SSDLC) integrates security activities into every phase of software development, ensuring that security is considered from initial concept through retirement. This section defines mandatory security activities, quality gates, and deliverables for each lifecycle phase.

4.1 SSDLC Overview

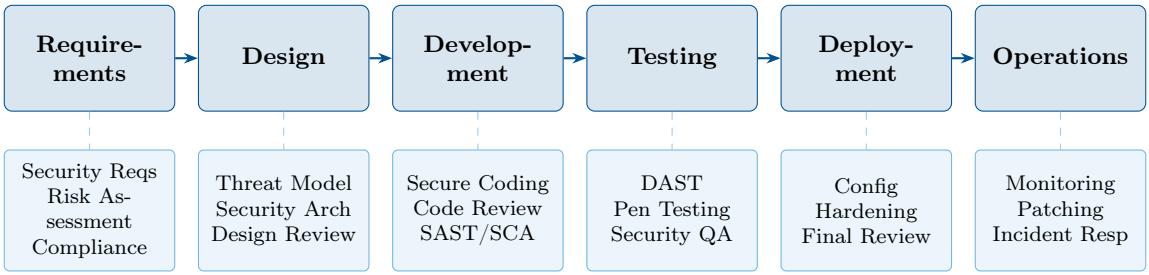


Figure 6: Secure Software Development Lifecycle with Security Activities

4.2 Requirements Phase

The requirements phase establishes security requirements alongside functional requirements, ensuring security considerations are addressed from project inception.

4.2.1 Security Requirements Identification

Security requirements derive from multiple sources:

- **Regulatory Requirements:** Mandates from applicable regulations such as data protection laws, industry-specific requirements, and international standards
- **Contractual Obligations:** Security commitments in customer contracts, service level agreements, and partnership arrangements
- **Organizational Policies:** Internal security policies, standards, and guidelines applicable to the application type and data classification
- **Threat Landscape:** Known threats and attack patterns relevant to the application's technology and business domain
- **Data Classification:** Requirements driven by the sensitivity of data the application will process, store, or transmit
- **Business Risk Assessment:** Security controls necessary to protect business value and maintain stakeholder confidence

4.2.2 Security Requirements Categories

Category	Example Requirements
Authentication	Multi-factor authentication for privileged access; session timeout after inactivity; password complexity requirements
Authorization	Role-based access control; principle of least privilege; segregation of duties for sensitive functions
Input Validation	Server-side validation of all input; parameterized queries; output encoding
Cryptography	Encryption of sensitive data at rest and in transit; approved algorithms and key lengths; key management procedures
Logging & Monitoring	Security event logging; audit trail integrity; alerting for suspicious activities
Error Handling	Secure error messages; exception handling without information leakage
Data Protection	Data minimization; retention limits; secure deletion; anonymization requirements
Session Management	Secure session token generation; session binding; timeout enforcement
Communication Sec.	TLS for all external communications; certificate validation; secure protocols only
Configuration	Secure defaults; hardening requirements; change management controls

Table 10: Security Requirements Categories

4.2.3 Requirements Phase Deliverables

Requirements Phase Security Deliverables

- Security requirements specification document
- Data classification determination
- Application risk tier classification
- Preliminary compliance requirements mapping
- Initial risk assessment
- Security acceptance criteria for user stories
- Abuse case documentation (misuse cases)

4.3 Design Phase

The design phase translates security requirements into secure architecture and design specifications. Security decisions made during design have the greatest leverage for preventing vulnerabilities cost-effectively.

4.3.1 Security Architecture Principles

All application designs should adhere to established security architecture principles:

Security Architecture Principles

1. **Defense in Depth:** Implement multiple layers of security controls so that failure of any single control does not compromise security
2. **Least Privilege:** Grant minimum permissions necessary for each component, user, and process to perform its intended function
3. **Secure Defaults:** Configure systems to be secure out of the box; require explicit action to reduce security
4. **Fail Secure:** Design systems to fail in a secure state; deny access when controls fail
5. **Separation of Duties:** Divide critical functions among multiple components or individuals to prevent single points of compromise
6. **Economy of Mechanism:** Keep security designs as simple as possible; complexity increases attack surface and error likelihood
7. **Complete Mediation:** Verify authorization for every access attempt; never rely on cached permissions
8. **Open Design:** Security should not depend on secrecy of design; assume adversaries know the system architecture
9. **Psychological Acceptability:** Security mechanisms should not impede legitimate use or users will circumvent them
10. **Weakest Link:** Security is only as strong as the weakest component; identify and strengthen weak points

4.3.2 Threat Modeling

Threat modeling systematically identifies potential threats to an application and determines appropriate countermeasures. All Tier 1 and Tier 2 applications require formal threat modeling during design.

Activity	Description
Scope Definition	Define the boundaries of analysis; identify assets, entry points, and trust boundaries
Decomposition	Create architectural diagrams showing components, data flows, and trust levels
Threat Identification	Systematically identify threats using structured approaches such as STRIDE or attack trees
Vulnerability Analysis	Identify potential vulnerabilities that could enable identified threats
Countermeasure Design	Specify controls to prevent, detect, or respond to identified threats
Risk Assessment	Evaluate residual risk after countermeasures; prioritize based on risk score
Documentation	Record threat model artifacts for future reference and updates

Table 11: Threat Modeling Process Steps

Threat Modeling Triggers

Beyond initial design, threat modeling should be revisited when:

- Significant architectural changes occur
- New data types or sensitivity levels are introduced
- External interfaces or integrations are added
- Trust boundaries are modified
- New threat intelligence indicates emerging attack patterns
- Major security incidents affect similar applications

4.3.3 Security Design Review

Security design reviews evaluate proposed architectures against security requirements and best practices. Review depth scales with application risk tier.

App Tier	Review Type	Requirements
Tier 1	Full security architecture review	Formal review by security architecture team; documented approval required before development
Tier 2	Standard design review	Review by Application Security team member; threat model validation
Tier 3	Lightweight review	Security Champion review with AppSec consultation as needed
Tier 4	Self-assessment	Development team completes security design checklist

Table 12: Design Review Requirements by Application Tier

4.3.4 Design Phase Deliverables

Design Phase Security Deliverables

- Security architecture document
- Threat model with identified threats and countermeasures
- Data flow diagrams with trust boundaries
- Security design review approval
- Updated risk assessment
- Authentication and authorization design
- Cryptographic design specification
- Security control specifications

4.4 Development Phase

The development phase implements the secure design through secure coding practices, code review, and automated security analysis. This phase has the highest volume of security activities but benefits from automation to maintain development velocity.

4.4.1 Secure Coding Practices

Developers must follow secure coding standards that address common vulnerability categories:

Vulnerability Category	Secure Coding Requirements
Injection	Use parameterized queries for all database access; validate and sanitize all input; use safe APIs that avoid shell interpretation; implement appropriate output encoding
Broken Authentication	Use proven authentication frameworks; implement proper password storage with modern hashing algorithms; enforce session management controls; protect credentials in transit and storage
Sensitive Data Exposure	Classify data appropriately; encrypt sensitive data at rest and in transit; minimize data collection and retention; implement proper key management
XML External Entities	Disable external entity processing; use less complex data formats where possible; validate and sanitize XML input
Broken Access Control	Implement server-side authorization checks; deny by default; validate permissions on every request; log access control failures
Security Misconfiguration	Follow hardening guides; remove unnecessary features; keep dependencies updated; use secure defaults
Cross-Site Scripting	Encode output appropriately for context; use security-focused templating engines; implement Content Security Policy; validate and sanitize input
Insecure Deserialization	Avoid deserializing untrusted data; implement integrity checks; use simple data formats; isolate deserialization code
Vulnerable Components	Maintain software bill of materials; monitor for vulnerabilities; update components promptly; evaluate component security before adoption
Insufficient Logging	Log security-relevant events; protect log integrity; include sufficient context; enable monitoring and alerting

4.4.2 Code Review

Code review provides human verification of security practices that automated tools may miss. Security-focused code review examines authentication and authorization implementation, cryptographic usage, input validation and output encoding, error handling and logging, business logic security, and compliance with secure coding standards.

All code changes to Tier 1 and Tier 2 applications require security-aware code review before merge. Code reviewers must complete secure code review training and use security-focused review checklists.

4.4.3 Static Application Security Testing (SAST)

Static analysis examines source code or compiled binaries to identify potential security vulnerabilities without executing the application. SAST integration requirements:

SAST Integration Requirements

- SAST scanning integrated into CI/CD pipeline for all repositories
- Scans execute on every code commit or pull request
- Critical and high severity findings block build progression for Tier 1 and Tier 2 applications
- Findings triaged within defined SLA based on severity
- False positives documented and suppressed with justification
- Custom rules implemented for organization-specific patterns
- Regular tuning to minimize false positives while maintaining detection

4.4.4 Software Composition Analysis (SCA)

SCA identifies security vulnerabilities and license compliance issues in third-party and open-source components. Given that modern applications typically consist of 70-90% third-party code, SCA is essential for comprehensive coverage.

SCA Capability	Requirements
Component Inventory	Automated discovery and inventory of all direct and transitive dependencies
Vulnerability Detection	Real-time matching against vulnerability databases with continuous monitoring
License Compliance	Identification of license types and flagging of incompatible or restricted licenses
Version Currency	Identification of outdated components with available updates
Risk Scoring	Risk assessment considering vulnerability severity, exploitability, and component usage
Remediation Guidance	Recommended versions, upgrade paths, and workarounds
SBOM Generation	Automated generation of software bill of materials in standard formats

Table 14: SCA Capability Requirements

4.4.5 Secrets Management

Credentials, API keys, certificates, and other secrets must never be committed to source code repositories. Development practices must include:

- Pre-commit hooks to detect potential secrets in code
- Secrets scanning integrated into CI/CD pipelines
- Centralized secrets management for all environments
- Environment-specific secrets injection at runtime
- Immediate rotation of any secrets detected in repositories
- Regular secrets inventory and rotation

4.4.6 Development Phase Deliverables

Development Phase Security Deliverables

- SAST scan reports with resolved findings
- SCA scan reports with resolved findings
- Code review records with security sign-off
- Updated software bill of materials
- Secrets scan verification
- Unit tests for security controls
- Security-relevant code documentation

4.5 Testing Phase

The testing phase validates that security controls function as designed through dynamic testing, penetration testing, and security-focused quality assurance.

4.5.1 Dynamic Application Security Testing (DAST)

DAST tests running applications by simulating attacks to identify runtime vulnerabilities. Unlike SAST, DAST finds vulnerabilities in deployed configurations and runtime behavior.

DAST Element	Requirements
Scan Frequency	Automated scans in CI/CD for every deployment to test environments; weekly scans of production applications
Coverage	Full crawl and scan of all application endpoints; authenticated scanning with appropriate credentials
Test Categories	OWASP Top 10 coverage; injection testing; authentication bypass; authorization testing; session management
Finding Handling	Automatic ticket creation for verified findings; integration with vulnerability management system
False Positive Management	Triage and documentation of false positives; tuning to improve accuracy
Environment Considerations	Scanning against representative test environments; production scanning during low-traffic periods

Table 15: DAST Program Requirements

4.5.2 Interactive Application Security Testing (IAST)

IAST combines elements of SAST and DAST by using agents deployed within the running application to monitor code execution during testing. IAST provides:

- Real-time vulnerability detection during functional testing
- Precise identification of vulnerable code locations
- Lower false positive rates than standalone SAST or DAST
- Coverage of data flow through the entire application stack
- Integration with existing functional test suites

IAST deployment is recommended for Tier 1 and Tier 2 applications where the additional accuracy and precision justify the integration effort.

4.5.3 Penetration Testing

Penetration testing simulates real-world attacks by skilled testers to identify vulnerabilities that automated tools may miss. The penetration testing program includes:

App Tier	Test Type	Frequency	Scope
Tier 1	Full penetration test	Annual minimum; after major changes	Complete application including infrastructure; red team exercises
Tier 2	Application penetration test	Annual minimum	Application layer focus; API testing
Tier 3	Targeted assessment	Every 2 years	Risk-based scope; specific functionality
Tier 4	On request	As warranted	Based on significant changes or concerns

Table 16: Penetration Testing Requirements by Application Tier

4.5.4 Security Test Case Development

Security test cases should be developed alongside functional test cases to verify security requirements and control implementations:

- Authentication test cases verifying proper credential handling and session management
- Authorization test cases confirming access control enforcement
- Input validation test cases with malicious input patterns
- Boundary condition tests for security-relevant functions
- Negative test cases verifying proper denial of unauthorized actions
- Regression tests for previously identified vulnerabilities

4.5.5 Testing Phase Deliverables

Testing Phase Security Deliverables

- DAST scan reports with resolved findings
- IAST analysis results (where applicable)
- Penetration test report with remediation status
- Security test case execution results
- Security regression test results
- Updated risk assessment with test findings
- Security sign-off for release

4.6 Deployment Phase

The deployment phase ensures secure configuration and hardening before applications enter production.

4.6.1 Security Configuration Management

All applications must be deployed with secure configurations verified against hardening baselines:

- Security configuration baselines documented for each technology stack
- Automated configuration scanning to detect deviations
- Change management controls for security-relevant configurations
- Separation of configuration from code
- Environment-specific security controls appropriately configured
- Removal of default credentials, sample data, and development artifacts

4.6.2 Deployment Security Checklist

Pre-Production Security Checklist

Code Security:
- All critical and high SAST findings remediated or accepted
- All critical and high SCA findings remediated or accepted
- Secrets scanning verified clean
- Security code review completed

Testing Security:
- DAST scan completed with findings addressed
- Penetration test completed (per tier requirements)
- Security test cases passed
- No known critical or high vulnerabilities

Configuration Security:
- Hardening baseline applied and verified
- TLS configuration validated
- Security headers configured
- Logging and monitoring enabled
- Access controls configured

Documentation:
- Security architecture documentation current
- Runbook includes security procedures
- Incident response procedures documented
- Risk acceptance documented for residual risks

4.6.3 Security Gate Approval

Deployment to production requires security gate approval based on application tier:

App Tier	Approval Authority	Requirements
Tier 1	Application Security Manager	All checklist items verified; penetration test current; formal sign-off
Tier 2	Application Security Team Lead	Security checklist completed; SAST/DAST/SCA clear
Tier 3	Security Champion	Automated scans passing; checklist self-assessment
Tier 4	Development Team Lead	Automated security gates passing

Table 17: Security Gate Approval Requirements

4.7 Operations and Maintenance Phase

Security activities continue throughout the operational life of applications, addressing emerging threats, newly discovered vulnerabilities, and ongoing compliance requirements.

4.7.1 Continuous Security Monitoring

Production applications require ongoing security monitoring:

- Runtime application self-protection (RASP) for Tier 1 applications
- Web application firewall (WAF) protection for internet-facing applications
- Security event logging with SIEM integration
- Anomaly detection for application behavior
- Regular vulnerability scanning of production systems
- Certificate expiration monitoring
- Dependency vulnerability monitoring

4.7.2 Patch and Update Management

Security patches and updates must be applied within defined timeframes:

Severity	Patch Timeline	Notes
Critical	Within 24-72 hours	Emergency change process; immediate for actively exploited
High	Within 7-14 days	Prioritized in current sprint
Medium	Within 30-60 days	Scheduled maintenance window
Low	Within 90 days	Normal release cycle

Table 18: Security Patch Timelines

4.7.3 Application Retirement

Applications reaching end-of-life require secure retirement procedures:

- Data retention requirements reviewed and implemented
- Sensitive data securely deleted or archived
- Credentials and secrets revoked
- Access permissions removed
- Application removed from inventory and monitoring
- Documentation archived
- Lessons learned captured

5 Security Testing Program

The Security Testing Program establishes comprehensive testing capabilities to identify vulnerabilities throughout the application lifecycle. This section provides detailed guidance on testing methodologies, tools, and processes.

5.1 Testing Program Objectives

The Security Testing Program pursues the following objectives:

1. Identify security vulnerabilities before they reach production
2. Verify effectiveness of security controls
3. Meet compliance requirements for security testing
4. Provide developers with timely, actionable feedback
5. Continuously improve security posture through testing insights
6. Measure and report on security quality metrics

5.2 Testing Coverage Model

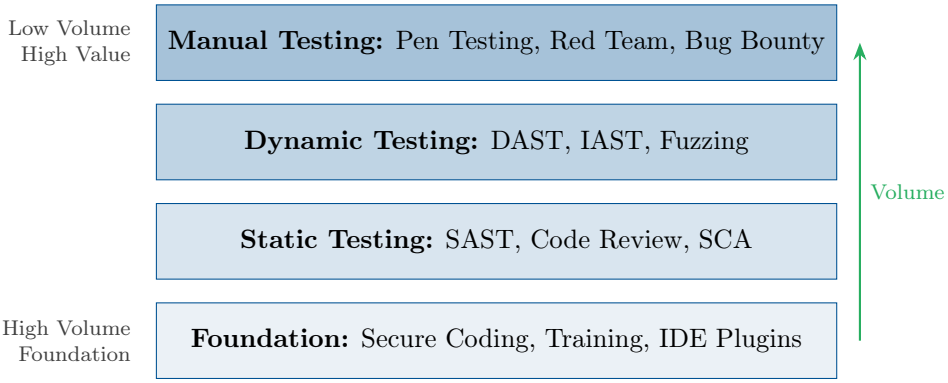


Figure 7: Security Testing Coverage Pyramid

5.3 Automated Security Testing

5.3.1 CI/CD Integration

Security testing must be integrated into continuous integration and continuous deployment pipelines to provide immediate feedback and enforce security gates.

Pipeline Stage	Security Tests	Gate Criteria
Pre-Commit	Secrets scanning, linting	Block commit if secrets detected
Build	SAST, SCA, container scanning	No critical findings; high findings reviewed
Test	DAST, IAST, security unit tests	No critical findings; test coverage met
Pre-Deploy	Configuration scanning, final SAST	All gates passed; approval obtained
Post-Deploy	Smoke tests, baseline validation	Security controls operational

Table 19: CI/CD Security Integration Points

5.3.2 Scan Scheduling and Triggers

Test Type	Trigger	Additional Schedule
SAST	Every commit/pull request	Full repository scan weekly
SCA	Every build	Continuous monitoring for new CVEs
Secrets Scan	Pre-commit hook; every build	Weekly full repository scan
DAST	Deploy to test environment	Weekly production scan
Container Scan	Every image build	Daily scan of deployed images
Config Scan	Every deployment	Daily compliance check

Table 20: Security Scan Scheduling

5.4 Manual Security Testing

5.4.1 Penetration Testing Methodology

Penetration tests follow a structured methodology regardless of internal or external execution:

1. **Scoping:** Define test boundaries, objectives, rules of engagement, and success criteria
2. **Reconnaissance:** Gather information about target applications and infrastructure
3. **Vulnerability Discovery:** Identify potential vulnerabilities through manual and automated techniques
4. **Exploitation:** Attempt to exploit identified vulnerabilities to demonstrate impact
5. **Post-Exploitation:** Assess potential for lateral movement and additional compromise
6. **Reporting:** Document findings with severity ratings, evidence, and remediation guidance
7. **Remediation Validation:** Verify that fixes effectively address identified vulnerabilities

5.4.2 Red Team Exercises

Red team exercises simulate sophisticated adversary attacks against the organization's applications and infrastructure. Unlike penetration testing, red team exercises:

- Focus on objectives rather than vulnerability enumeration
- Test detection and response capabilities alongside prevention
- Use stealth and evasion techniques
- May span extended timeframes
- Simulate realistic threat actor tactics, techniques, and procedures

Red team exercises are recommended annually for Tier 1 applications and critical infrastructure.

5.4.3 Bug Bounty Program

Bug bounty programs leverage external security researchers to identify vulnerabilities. Program structure:

Bug Bounty Program Structure

Scope Definition:
- In-scope applications and domains
- Out-of-scope areas and prohibited activities
- Qualifying vulnerability types
- Testing guidelines and rules of engagement

Reward Structure:
- Tiered rewards based on severity and impact
- Clear criteria for reward determination
- Bonus criteria for exceptional findings
- Timeline for reward payment

Operational Process:
- Submission intake and triage process
- Response time commitments
- Communication protocols with researchers
- Disclosure coordination procedures

5.5 Vulnerability Severity Classification

All security findings are classified using a consistent severity framework:

Severity	CVSS Range	Description and Examples
Critical	9.0 – 10.0	Immediate, severe impact; trivial exploitation. Examples: unauthenticated RCE, SQL injection with full database access, authentication bypass to admin
High	7.0 – 8.9	Significant impact; exploitation likely. Examples: stored XSS affecting multiple users, privilege escalation, significant data exposure
Medium	4.0 – 6.9	Moderate impact; exploitation requires conditions. Examples: reflected XSS, CSRF, information disclosure of moderate sensitivity
Low	0.1 – 3.9	Limited impact; difficult exploitation. Examples: verbose error messages, minor information disclosure, missing security headers
Info	N/A	Best practice recommendations; no direct security impact

Table 21: Vulnerability Severity Classification

6 Vulnerability Management

Vulnerability management encompasses the processes for identifying, assessing, prioritizing, remediating, and reporting security vulnerabilities across the application portfolio. Effective vulnerability management is essential for reducing organizational risk and demonstrating due diligence.

6.1 Vulnerability Management Lifecycle

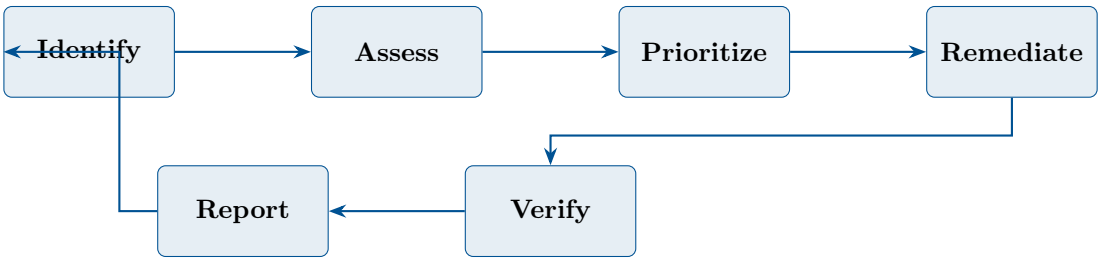


Figure 8: Vulnerability Management Lifecycle

6.2 Vulnerability Identification

Vulnerabilities are identified through multiple channels:

Source	Description
Automated Scanning	SAST, DAST, SCA, IAST, infrastructure scanning
Manual Testing	Penetration testing, code review, security assessments
External Research	Bug bounty submissions, responsible disclosure
Threat Intelligence	Vendor advisories, CVE publications, security bulletins
Internal Discovery	Developer reports, operations team findings, incident investigations
Third-Party Assessments	Customer security assessments, audit findings, compliance reviews

Table 22: Vulnerability Identification Sources

6.3 Vulnerability Assessment and Prioritization

Raw vulnerability data must be assessed and prioritized to focus remediation efforts on the highest-risk issues.

6.3.1 Assessment Factors

Factor	Considerations
Technical Severity	CVSS base score; attack complexity; required privileges
Exploitability	Exploit availability; active exploitation; ease of exploitation
Asset Criticality	Application tier; data sensitivity; business impact
Exposure	Internet-facing vs. internal; network accessibility; attack surface
Compensating Controls	Existing mitigations; defense in depth; monitoring coverage
Business Context	Regulatory implications; customer impact; reputational risk

Table 23: Vulnerability Assessment Factors

6.3.2 Prioritization Model

The organization uses a risk-based prioritization model that considers both technical severity and business context:

$$\text{Priority Score} = \text{Technical Severity} \times \text{Asset Criticality} \times \text{Exposure Factor}$$

Priority	Score Range	Treatment
P1	75 – 100	Immediate response; emergency remediation
P2	50 – 74	Urgent; next sprint/release
P3	25 – 49	Standard; scheduled remediation
P4	1 – 24	Low; address as capacity permits

Table 24: Vulnerability Priority Levels

6.4 Remediation Requirements

6.4.1 Service Level Agreements

Vulnerability remediation must occur within defined SLA timeframes based on priority:

Remediation SLA Requirements

Priority	Internet-Facing	Internal
P1 (Critical)	24 – 72 hours	7 days
P2 (High)	7 days	14 days
P3 (Medium)	30 days	60 days
P4 (Low)	90 days	180 days

SLA timers begin when a vulnerability is confirmed and assigned. Extensions require documented approval according to the exception management process.

6.4.2 Remediation Options

Multiple approaches exist for addressing vulnerabilities:

1. **Fix:** Correct the underlying vulnerability through code or configuration change (preferred)
2. **Patch:** Apply vendor-supplied security patch
3. **Upgrade:** Update to a version where the vulnerability is addressed
4. **Mitigate:** Implement compensating controls that reduce risk without fixing root cause
5. **Accept:** Document acceptance of residual risk (requires approval per exception process)
6. **Retire:** Decommission the vulnerable component or application

6.5 Verification and Closure

All remediated vulnerabilities require verification before closure:

- Automated rescan confirming vulnerability no longer detected
- Manual verification for complex vulnerabilities
- Regression testing to ensure fix does not introduce new issues
- Documentation of remediation approach
- Update of tracking system with closure evidence

6.6 Vulnerability Tracking and Reporting

6.6.1 Tracking Requirements

All vulnerabilities are tracked in a centralized vulnerability management system with:

- Unique identifier for each vulnerability
- Source, discovery date, and discoverer
- Affected application and component
- Technical details and evidence
- Severity and priority classification
- Assigned owner and remediation team
- SLA due date and current status
- Remediation history and closure evidence

6.6.2 Reporting Cadence

Report	Frequency	Audience
Operational Dashboard	Real-time	Application Security team
Development Team Report	Weekly	Development teams
Management Summary	Monthly	IT leadership
Executive Report	Quarterly	Steering Committee, Executives
Compliance Report	Per requirement	Auditors, Regulators

Table 25: Vulnerability Reporting Cadence

7 Third-Party and Supply Chain Security

Modern applications extensively rely on third-party code, including commercial software, open-source libraries, cloud services, and external APIs. This section establishes requirements for managing security risks throughout the software supply chain.

7.1 Supply Chain Risk Landscape

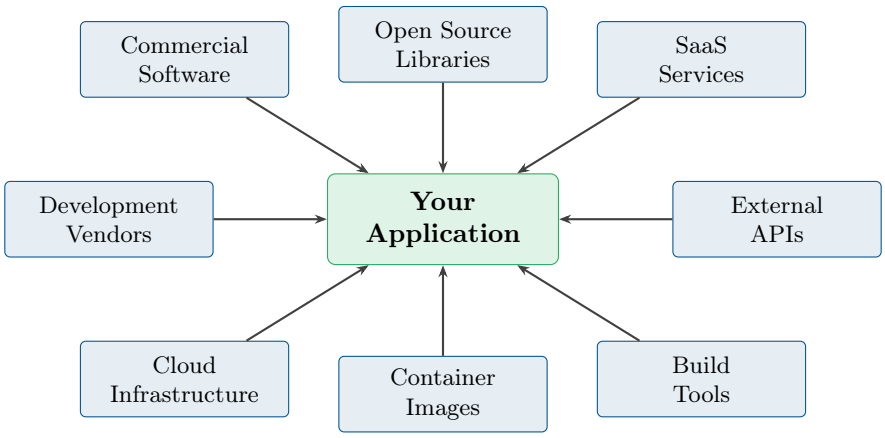


Figure 9: Software Supply Chain Components

7.2 Vendor Security Assessment

7.2.1 Assessment Requirements

Third-party vendors providing software, services, or access to organizational data must undergo security assessment proportionate to risk:

Risk Tier	Criteria	Assessment Requirements
Critical	Processes sensitive data; critical infrastructure; high access	Full security assessment; on-site review option; contractual requirements; annual reassessment
High	Moderate data access; business-important service; integration with internal systems	Security questionnaire; evidence review; contractual requirements; reassessment every 2 years
Medium	Limited data access; replaceable service; minimal integration	Abbreviated questionnaire; certification review; standard terms
Low	No data access; commodity service; no integration	Self-attestation; standard terms

Table 26: Vendor Security Assessment Tiers

7.2.2 Assessment Process

1. **Risk Classification:** Determine vendor risk tier based on data access, integration, and business criticality
2. **Questionnaire:** Distribute appropriate security questionnaire
3. **Evidence Review:** Examine provided certifications, audit reports, and policies
4. **Gap Analysis:** Identify gaps between vendor capabilities and organizational requirements
5. **Risk Decision:** Accept, require remediation, or reject based on assessment results
6. **Contractual Controls:** Negotiate appropriate security terms and requirements
7. **Ongoing Monitoring:** Track vendor security posture throughout relationship

7.3 Open Source Security

7.3.1 Open Source Governance

Open source usage requires governance to manage security, license, and quality risks:

Open Source Governance Requirements

Pre-Adoption Review:
- Security history and vulnerability record
- Maintenance activity and community health
- License compatibility with intended use
- Code quality indicators
- Available alternatives comparison

Ongoing Management:
- Continuous vulnerability monitoring via SCA
- Version currency tracking
- License change monitoring
- Maintainer and ownership changes
- End-of-life and abandonment detection

Approved/Prohibited Lists:
- Maintain list of pre-approved components
- Document prohibited components with rationale
- Establish review process for new component requests

7.3.2 Dependency Management

Effective dependency management minimizes supply chain attack surface:

- Pin dependency versions in production configurations
- Use lock files to ensure reproducible builds
- Prefer well-maintained dependencies with active security response
- Minimize dependency depth where practical
- Regularly audit for unused dependencies
- Implement dependency confusion protections
- Verify package integrity through checksums or signatures

7.4 Software Bill of Materials (SBOM)

7.4.1 SBOM Requirements

All applications must maintain current Software Bills of Materials documenting their components:

SBOM Element	Requirements
Format	Standard format (SPDX, CycloneDX, or SWID)
Content	All direct and transitive dependencies; version information; supplier; license
Generation	Automated generation during build process
Currency	Updated with every release
Storage	Retained in accessible repository with versioning
Sharing	Available for customer requests and regulatory requirements
Analysis	Integrated with vulnerability monitoring for continuous risk assessment

Table 27: SBOM Requirements

7.5 Build Pipeline Security

Securing the build pipeline prevents supply chain attacks that compromise the software delivery process:

- Secure configuration of CI/CD systems
- Access controls and audit logging for pipeline modifications
- Signed commits and verified contributor identity
- Artifact signing and verification
- Isolated build environments
- Dependency verification during build
- Pipeline-as-code with version control
- Regular security review of build configurations

8 Security Training and Awareness

Security training and awareness programs build the human capabilities necessary for effective application security. This section defines requirements for role-based training, developer education, and security awareness.

8.1 Training Program Objectives

1. Ensure all personnel understand their security responsibilities
2. Equip developers with skills to write secure code
3. Build specialized expertise within the security team
4. Create a security-conscious organizational culture
5. Meet compliance requirements for security training
6. Reduce human-factor security incidents

8.2 Role-Based Training Requirements

Role	Required Training	Frequency	Duration
All Employees	Security Awareness Fundamentals	Annual	1 hour
Developers	Secure Coding Foundations	Annual	4 hours
Developers	Language/Framework-Specific Security	At adoption	2-4 hours
Security Champions	Advanced Application Security	Annual	16+ hours
Security Team	Specialized Technical Training	Ongoing	Per plan
Architects	Security Architecture Principles	Annual	8 hours
QA Engineers	Security Testing Fundamentals	Annual	4 hours
Operations	Secure Operations and Monitoring	Annual	4 hours
Management	Security Leadership	Annual	2 hours
Executives	Executive Security Briefing	Annual	1 hour

8.3 Developer Security Training

8.3.1 Secure Coding Curriculum

Developer security training covers:

Developer Security Training Curriculum

Foundations:
- Security mindset and threat landscape
- Common vulnerability types and root causes
- Secure development lifecycle overview
- Security requirements and acceptance criteria

Defensive Coding:
- Input validation and output encoding
- Authentication and session management
- Access control implementation
- Cryptography fundamentals and proper usage
- Error handling and logging
- Secure configuration

Technology-Specific:
- Language-specific security features and pitfalls
- Framework security controls
- API security
- Cloud-native security considerations

Practical Skills:
- Using security testing tools
- Interpreting and triaging security findings
- Threat modeling participation
- Security code review

8.3.2 Training Delivery Methods

Multiple delivery methods accommodate different learning styles and schedules:

- Online self-paced modules for foundational content
- Instructor-led sessions for complex topics and hands-on exercises
- Capture-the-flag (CTF) exercises for practical skill building
- Lunch-and-learn sessions for topical updates
- Embedded learning through IDE plugins and code review feedback
- Secure coding challenges and competitions
- Post-incident training based on real events

8.4 Security Champion Program Training

Security Champions receive enhanced training to serve as security liaisons within development teams:

Track	Topics	Hours
Foundation	Advanced secure coding; threat modeling; security architecture principles	8
Tools	In-depth security tool training; finding interpretation; triage techniques	8
Process	Security requirements; risk assessment; exception handling; metrics	4
Leadership	Influencing without authority; security communication; team coaching	4

Table 29: Security Champion Training Tracks

8.5 Training Effectiveness Measurement

Training effectiveness is measured through:

- Completion rates and compliance tracking
- Knowledge assessments and certification examinations
- Practical skill demonstrations
- Vulnerability trends in trained teams' code
- Security Champion activity metrics
- Post-training surveys and feedback
- Correlation between training and security metrics

9 Application Security Incident Response

Application security incident response addresses security events affecting applications, including exploitation of vulnerabilities, data breaches through application vectors, and unauthorized access via application weaknesses.

9.1 Incident Response Objectives

1. Minimize impact of application security incidents
2. Restore secure application operations quickly
3. Preserve evidence for investigation and potential legal proceedings
4. Identify root causes and prevent recurrence
5. Meet notification requirements for affected parties
6. Capture lessons learned for program improvement

9.2 Incident Classification

Application security incidents are classified by severity to determine response urgency and escalation:

Severity	Criteria	Response Requirement
Critical	Active exploitation; significant data breach; critical system compromise	Immediate response; executive notification; war room activation
High	Likely exploitation; limited breach; high-value system affected	Response within 4 hours; management notification
Medium	Potential exploitation; suspicious activity; moderate impact	Response within 24 hours; standard process
Low	Minor security event; no evidence of exploitation; limited impact	Response within 72 hours; normal workflow

Table 30: Incident Severity Classification

9.3 Incident Response Process

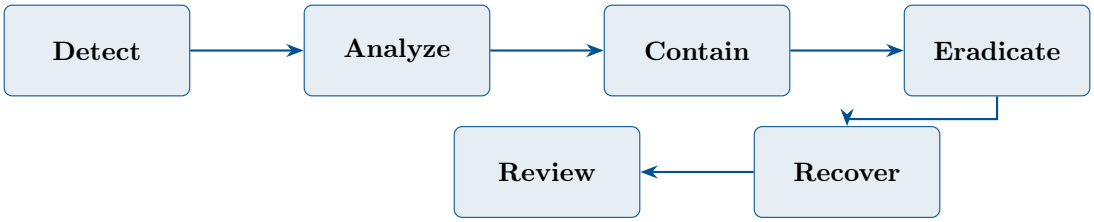


Figure 10: Incident Response Phases

9.3.1 Detection

Detection identifies potential application security incidents through:

- Security monitoring and alerting systems
- Web application firewall events
- Runtime application protection alerts
- Anomaly detection systems
- User reports and help desk tickets
- External notifications (researchers, partners, customers)
- Penetration testing and security assessment discoveries

9.3.2 Analysis

Analysis determines incident scope, impact, and root cause:

- Confirm whether a security incident has occurred
- Identify affected systems, data, and users
- Determine attack vector and exploitation method
- Assess data exposure and exfiltration
- Identify indicators of compromise
- Establish incident timeline
- Preserve evidence for investigation

9.3.3 Containment

Containment limits incident impact and prevents spread:

- Isolate affected systems if necessary
- Block malicious traffic and attack sources
- Disable compromised accounts
- Implement emergency patches or configuration changes
- Activate additional monitoring
- Communicate with stakeholders

9.3.4 Eradication

Eradication removes the threat and addresses root cause:

- Remove malicious code or backdoors
- Patch exploited vulnerabilities
- Reset compromised credentials
- Update security controls
- Verify complete removal of threat

9.3.5 Recovery

Recovery restores normal operations:

- Restore systems from known-good state
- Verify system integrity before return to service
- Implement enhanced monitoring during recovery period
- Communicate recovery status to stakeholders
- Validate security controls functioning correctly

9.3.6 Post-Incident Review

Post-incident review captures lessons learned and improves future response:

- Conduct formal post-incident review meeting
- Document incident timeline and response actions
- Identify what worked well and areas for improvement
- Determine root causes and contributing factors
- Define corrective actions with owners and deadlines
- Update incident response procedures as needed
- Share sanitized lessons learned across organization

9.4 Communication Requirements

Stakeholder	Communication Requirements
Exec. Leadership	Immediate notification for critical incidents; regular updates until resolved
Legal/Compliance	Notification when data breach suspected; guidance on regulatory requirements
Communications	Coordinate external messaging; prepare statements if needed
Affected Customers	Notification per regulatory requirements and contractual obligations
Regulators	Report within required timeframes based on jurisdiction and data type
Business Units	Keep informed of impact and recovery status

Table 31: Incident Communication Requirements

10 Compliance and Audit

The Application Security Program supports organizational compliance with regulatory requirements, industry standards, and contractual obligations through documented controls, regular assessments, and audit-ready evidence.

10.1 Compliance Framework

10.1.1 Regulatory Requirements

The organization identifies and tracks applicable regulations affecting application security:

Regulation Type	Examples and Considerations
Data Protection	General data protection regulations; privacy laws; cross-border transfer requirements
Financial Services	Payment card security; financial reporting controls; trading system requirements
Healthcare	Protected health information; medical device security; healthcare interoperability
Government	Federal information security; defense contractor requirements; critical infrastructure protection
Industry-Specific	Telecommunications; energy sector; transportation security

Table 32: Regulatory Requirement Categories

10.1.2 Industry Standards

The program aligns with recognized industry standards and frameworks:

- OWASP Application Security Verification Standard (ASVS)
- NIST Secure Software Development Framework (SSDF)
- ISO/IEC 27034 Application Security
- Payment Card Industry Data Security Standard (PCI DSS)
- SOC 2 Trust Services Criteria
- Cloud Security Alliance controls

10.2 Control Documentation

10.2.1 Control Library

The Application Security Program maintains a control library documenting:

- Control identifier and name
- Control objective and description
- Implementation guidance
- Testing procedures
- Evidence requirements
- Control owner
- Mapping to compliance requirements
- Related policies and procedures

10.2.2 Evidence Management

Compliance evidence must be systematically collected, retained, and accessible:

Evidence Management Requirements

Evidence Types:

- Security scan reports (SAST, DAST, SCA)
- Penetration test reports
- Training completion records
- Policy acknowledgments
- Risk assessment documentation
- Exception records and approvals
- Incident response records
- Change management records

Retention Requirements:

- Retain evidence for compliance period plus buffer (typically 3-7 years)
- Ensure evidence integrity and authenticity
- Maintain evidence accessibility for audit requests
- Protect evidence confidentiality appropriately

10.3 Audit Support

10.3.1 Internal Audit

Internal audit periodically assesses Application Security Program effectiveness:

- Annual program audit covering governance, processes, and controls
- Targeted audits of specific program areas
- Follow-up on previous audit findings
- Coordination with enterprise internal audit function

10.3.2 External Audit

External audits from regulators, customers, and certification bodies require coordinated support:

- Designated audit liaison within Application Security team
- Pre-audit preparation and evidence compilation
- Timely response to auditor requests
- Tracking and remediation of audit findings
- Management response documentation

10.4 Compliance Monitoring

Continuous compliance monitoring identifies deviations before they become audit findings:

- Automated control testing where possible
- Regular self-assessments against control requirements
- Compliance dashboards tracking key indicators
- Trend analysis identifying degrading compliance
- Integration with vulnerability and configuration management

11 Metrics and Reporting

Metrics and reporting enable data-driven management of the Application Security Program by measuring performance, identifying trends, and communicating status to stakeholders.

11.1 Metrics Framework

11.1.1 Metrics Categories

Category	Purpose	Example Metrics
Coverage	Measure program reach and adoption	Percentage of applications with SAST; threat model coverage; training completion rate
Effectiveness	Assess control and process quality	Vulnerability escape rate; mean time to remediate; finding recurrence rate
Efficiency	Evaluate resource utilization	Security review cycle time; cost per vulnerability found; automation rate
Risk	Quantify security posture	Open critical vulnerabilities; risk score trends; exploited vulnerability count
Compliance	Track regulatory adherence	Compliance percentage; audit finding count; exception volume

Table 33: Security Metrics Categories

11.2 Key Performance Indicators

11.2.1 Primary KPIs

Metric	Definition	Target
SAST Coverage	Percentage of repositories with SAST integration	100% for Tier 1-2; 90% overall
DAST Coverage	Percentage of web applications with DAST scanning	100% for Tier 1-2
SCA Coverage	Percentage of applications with dependency scanning	100%
Critical Vuln MTTR	Mean time to remediate critical vulnerabilities	$<$ 72 hours
High Vuln MTTR	Mean time to remediate high vulnerabilities	$<$ 14 days
Vulnerability SLA Compliance	Percentage of vulnerabilities remediated within SLA	$>$ 95%
Security Training Completion	Percentage of required training completed on time	$>$ 95%
Threat Model Coverage	Percentage of Tier 1-2 applications with current threat models	100%
Security Gate Pass Rate	Percentage of releases passing security gates on first attempt	$>$ 80%
Finding Recurrence Rate	Percentage of vulnerabilities that recur after remediation	$<$ 5%

11.2.2 Key Risk Indicators

Indicator	Risk Signal	Threshold
Aging Critical Vulnerabilities	Critical vulnerabilities open beyond SLA	$>$ 0
Vulnerability Backlog Growth	Month-over-month increase in open vulnerabilities	$>$ 10%
Security Exception Volume	Active security exceptions	$>$ 20
Failed Security Gates	Releases blocked by security findings	$>$ 25%
Unscanned Applications	Applications without recent security scanning	$>$ 5%
Overdue Penetration Tests	Applications past due for penetration testing	$>$ 0

Table 35: Key Risk Indicators

11.3 Reporting Structure

11.3.1 Operational Reporting

Operational reports support day-to-day program management:

- Real-time dashboards showing current vulnerability status
- Daily scan result summaries
- SLA tracking and approaching deadline alerts
- Tool health and coverage reports
- Development team security scorecards

11.3.2 Management Reporting

Management reports provide oversight visibility:

- Monthly program status reports
- Trend analysis and quarter-over-quarter comparisons
- Resource utilization and capacity reports
- Project status for security initiatives
- Risk register updates

11.3.3 Executive Reporting

Executive reports communicate strategic status:

- Quarterly security posture summaries
- Key risk indicators and trends
- Significant incidents and responses
- Program maturity progress
- Budget and resource needs
- Strategic initiative status

11.4 Continuous Improvement

Metrics drive continuous improvement through:

1. Regular review of metric trends to identify improvement opportunities
2. Root cause analysis of negative trends
3. Action planning and tracking for improvements
4. Benchmark comparison against industry data
5. Program retrospectives and lessons learned
6. Feedback collection from stakeholders
7. Annual program assessment and roadmap updates

12 Program Maturity

Program maturity assessment provides a framework for understanding current capabilities, setting improvement priorities, and tracking progress over time.

12.1 Maturity Model

The Application Security Program maturity model defines five levels of capability:

Level 5: Optimizing — Continuous improvement via metrics; predictive capabilities; industry leadership

Level 4: Managed — Quantitative measurement; consistent processes organization-wide; proactive risk management

Level 3: Defined — Standardized documented processes; security integrated into SDLC; dedicated resources

Level 2: Developing — Basic security tools deployed; some processes documented; reactive practices

Level 1: Initial — Ad-hoc security activities; no formal program; minimal tooling

Figure 11: Application Security Program Maturity Levels

12.2 Maturity Domains

Maturity is assessed across multiple domains:

Domain	Assessment Areas
Governance	Strategy; policy; organizational structure; accountability; risk management
Secure Development	Requirements; design; coding practices; code review; developer tooling
Security Testing	SAST; DAST; SCA; penetration testing; test coverage; automation
Vulnerability Mgmt	Identification; prioritization; remediation; verification; tracking
Third-Party Security	Vendor assessment; open source governance; supply chain security; SBOM
Training	Role-based training; developer education; security champions; awareness
Incident Response	Detection; analysis; response; recovery; post-incident improvement
Metrics & Reporting	Data collection; analysis; reporting; decision support

Table 36: Maturity Assessment Domains

12.3 Maturity Assessment Process

1. **Self-Assessment:** Program team evaluates current state against maturity criteria
2. **Evidence Review:** Validate self-assessment against documented evidence
3. **Stakeholder Input:** Gather perspectives from development teams, operations, and management
4. **Gap Analysis:** Identify gaps between current and target maturity levels
5. **Roadmap Development:** Define initiatives to address gaps with priorities and timelines
6. **Progress Tracking:** Monitor implementation and reassess periodically

12.4 Maturity Improvement Roadmap

Sample Maturity Improvement Roadmap

Year 1: Foundation (Target: Level 2→3)

- Establish formal governance structure and policies
- Deploy SAST and SCA across priority applications
- Implement vulnerability management process
- Launch developer security training program
- Define and begin tracking core metrics

Year 2: Standardization (Target: Level 3)

- Achieve full SAST/SCA coverage for Tier 1-2 applications
- Integrate security testing into CI/CD pipelines
- Establish Security Champion program
- Implement threat modeling for critical applications
- Automate compliance evidence collection

Year 3: Optimization (Target: Level 3→4)

- Expand coverage to all applications
- Implement advanced testing (IAST, RASP)
- Mature metrics and reporting capabilities
- Establish continuous improvement processes
- Achieve consistent process execution organization-wide

13 Conclusion

This Application Security Program provides the comprehensive framework necessary to systematically reduce application security risk across the enterprise. Success requires sustained commitment from executive leadership, adequate resource allocation, and cultural change that positions security as a shared responsibility rather than an obstacle to delivery.

Program Success Factors:
1. Executive Commitment: Visible leadership support, adequate resource allocation, and organizational authority for security requirements.
2. Integration Over Addition: Security activities embedded into existing workflows rather than bolted on as separate gates.
3. Risk-Based Approach: Focus resources on highest-risk applications and vulnerabilities rather than attempting uniform coverage.
4. Developer Experience: Security tools and processes that support rather than obstruct development productivity.
5. Measurable Progress: Clear metrics that demonstrate value and identify improvement opportunities.
6. Continuous Improvement: Regular assessment and evolution of program capabilities in response to changing threats and organizational needs.
7. Cultural Change: Building security awareness and capability throughout the organization, not just within a dedicated security team.

Implementation of this program should be phased based on organizational readiness and priorities. Starting with governance foundations, critical application coverage, and developer enablement provides the base for progressive maturity improvement.

Application security is not a destination but a journey.

The goal is not perfection but continuous improvement—
systematically reducing risk while enabling the business to innovate.

A Appendix A: Security Requirements Checklist

Category	Requirements
Authentication	Multi-factor authentication for sensitive functions; secure credential storage; account lockout after failed attempts; secure password reset; session timeout enforcement
Authorization	Role-based access control; least privilege implementation; authorization checks on all requests; segregation of duties for critical functions
Input Validation	Server-side validation for all input; whitelist validation where possible; parameterized database queries; proper encoding of output
Cryptography	Encryption of sensitive data at rest and in transit; approved algorithms only; proper key management; certificate validation
Error Handling	Secure error messages without sensitive information; comprehensive exception handling; appropriate logging of errors
Logging	Security event logging; log integrity protection; sufficient detail for investigation; log retention per policy
Session Mgmt	Secure session token generation; session binding to client; timeout enforcement; secure session termination
Data Protection	Data classification applied; encryption per classification; data minimization; secure deletion
Communication	TLS 1.2+ for all external communication; certificate validation; secure protocol configuration
Configuration	Secure defaults; hardening applied; removal of unnecessary features; change management

B Appendix B: Threat Modeling Template

Section	Content
Application Overview	Name, purpose, owner, risk tier, technology stack
Architecture Desc.	Components, data flows, external dependencies, trust boundaries
Assets	Sensitive data, critical functions, valuable resources
Entry Points	External interfaces, APIs, user inputs, file uploads
Threat Actors	Relevant threat actors and their capabilities/motivations
Threats Identified	Threat ID, description, STRIDE category, affected assets
Vulnerabilities	Potential weaknesses that could enable threats
Countermeasures	Existing and proposed controls for each threat
Risk Assessment	Likelihood, impact, and priority for each threat
Residual Risk	Remaining risk after countermeasures
Action Items	Required actions, owners, and target dates

Table 38: Threat Model Document Template

C Appendix C: Vulnerability Report Template

Field	Description
Vulnerability ID	Unique identifier
Title	Brief descriptive title
Discovery Date	Date vulnerability was identified
Source	How the vulnerability was discovered
Affected Application	Application name and version
Affected Component	Specific module, file, or function
Vulnerability Type	Classification (e.g., SQL injection, XSS)
Description	Detailed technical description
Proof of Concept	Steps to reproduce; evidence
Impact Assessment	Potential business and technical impact
CVSS Score	Common Vulnerability Scoring System rating
Severity	Critical/High/Medium/Low classification
Priority	Remediation priority assignment
Remediation Guidance	Recommended fix approach
Assigned Owner	Person responsible for remediation
SLA Due Date	Remediation deadline
Status	Current status in remediation workflow
Resolution	How the vulnerability was addressed
Verification	Evidence that fix is effective

Table 39: Vulnerability Report Template

D Appendix D: Security Tool Categories

Acro.	Full Name	Description
SAST	Static App Sec Testing	Analyzes source code or binaries for vulnerabilities without execution
DAST	Dynamic App Sec Testing	Tests running applications by simulating attacks
IAST	Interactive App Sec Testing	Combines static and dynamic analysis using runtime agents
SCA	Software Composition Analysis	Identifies vulnerabilities in third-party components
RASP	Runtime App Self-Protection	Provides runtime protection from within the application
WAF	Web App Firewall	Filters and monitors HTTP traffic to protect web applications
ASPM	App Security Posture Mgmt	Unified visibility and risk management across app security
SBOM	Software Bill of Materials	Inventory of all software components
AST	App Security Testing	General category encompassing SAST, DAST, IAST
CSPM	Cloud Security Posture Mgmt	Monitors cloud configurations for security issues

E Appendix E: Reference Standards and Frameworks

- **OWASP ASVS** — Application Security Verification Standard
- **OWASP SAMM** — Software Assurance Maturity Model
- **NIST SSDF** — Secure Software Development Framework (SP 800-218)
- **NIST CSF** — Cybersecurity Framework
- **ISO/IEC 27034** — Application Security
- **BSIMM** — Building Security In Maturity Model
- **CIS Controls** — Center for Internet Security Controls
- **MITRE ATT&CK** — Adversarial Tactics, Techniques, and Common Knowledge
- **STRIDE** — Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege
- **CVSS** — Common Vulnerability Scoring System