

Secure Development Guidelines

Policy Statement

All software applications and systems developed or customized by World Bank, or by third parties on behalf of World Bank, must strictly adhere to a Secure Software Development Lifecycle (SSDLC). This policy mandates the integration of security considerations, activities, and controls throughout all phases of software development, from initial requirements gathering to deployment and ongoing maintenance. The objective is to minimize security vulnerabilities, protect Bank data and systems from compromise, and ensure the confidentiality, integrity, and availability of our financial services.

Scope

This policy applies to all software development and acquisition activities within World Bank. This includes, but is not limited to:

- Internal development of new applications (web, mobile, desktop).
- Significant enhancements or modifications to existing in-house applications.
- Customization of Commercial Off-The-Shelf (COTS) software.
- Development and integration of Application Programming Interfaces (APIs).
- Software developed by third-party vendors specifically for World Bank.
- The use of open-source components within any developed software.

Roles and Responsibilities

A collaborative approach is essential for the successful implementation of secure development practices :

- **Development Teams (Internal and External Vendors):** Primarily responsible for understanding and implementing secure coding standards, conducting unit-level security testing, remediating identified vulnerabilities in their code, and actively participating in security design reviews and threat modeling sessions.
- **Security Champions:** Designated individuals within development teams who act as local security advocates and subject matter experts. They promote secure development practices, provide initial security guidance to their peers, and serve as the primary liaison with the central Cybersecurity Team for security-related matters within their projects.
- **Cybersecurity Team:** Responsible for establishing and maintaining the Secure Development Guidelines, providing comprehensive security training to developers, defining security requirements for new projects, performing independent security

testing (e.g., SAST, DAST, penetration testing), conducting security architecture reviews, and approving security-critical design decisions.

- **Product Owners/Managers:** Accountable for integrating security requirements into product backlogs, user stories, and acceptance criteria. They must ensure that security considerations are prioritized alongside functional requirements throughout the development lifecycle.
- **Architects (Solution/Enterprise):** Responsible for designing secure application and system architectures, ensuring that designs incorporate principles like defense-in-depth, secure defaults, and appropriate segregation of duties.
- **Quality Assurance (QA) Teams:** Responsible for incorporating security test cases into their functional and non-functional testing efforts, verifying that security requirements have been met.

Secure SDLC Phases and Activities

Security is to be embedded into each phase of the Software Development Lifecycle (SDLC) :

1. Requirements Phase:

- **Security Requirements Definition:** Non-functional security requirements (e.g., authentication strength, encryption standards, audit logging levels, input validation rules) and functional security requirements (e.g., specific access control mechanisms, fraud detection features) must be explicitly defined, documented, and prioritized alongside business and functional requirements at the project's inception. These requirements are derived from World Bank's security policies, regulatory obligations (e.g., FFIEC, GDPR, PCI DSS if applicable), and risk assessments.
- **Data Classification:** The types of data the application will handle (e.g., public, internal, confidential, restricted financial data, PII) must be identified and classified. This classification dictates the level of security controls required for data protection (e.g., encryption strength, access restrictions, audit logging intensity).
- **Compliance Identification:** All applicable legal, regulatory, and contractual security obligations (e.g., data residency requirements for certain jurisdictions, specific FFIEC cybersecurity mandates) must be identified and incorporated as requirements.
- **Abuse Case Definition:** Potential misuse or abuse scenarios (e.g., unauthorized fund transfer attempts, fraudulent account opening) should be identified and documented to inform security design and testing.

- *Security Manager's Perspective*: For any new World Bank application intended to process or store customer financial information or facilitate financial transactions, a mandatory "Security Requirements Elicitation and Data Protection Impact Assessment (DPIA)" workshop is conducted. This involves representatives from Business, Legal, Compliance, Development, and Cybersecurity to ensure all security and privacy requirements are comprehensively captured and understood from the outset, aligning with a "shift-left" security philosophy.

2. Design Phase:

- **Threat Modeling**: Threat modeling must be performed for all new applications and for any significant architectural changes to existing critical applications. Methodologies such as STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) are employed to systematically identify potential threats, vulnerabilities, attack vectors, and define appropriate mitigations. Outputs of threat modeling (e.g., identified threats, required countermeasures) must be documented and integrated into the design and development backlog.
 - *Use Case (Threat Modeling for Financial Application)*: During the design of a new World Bank peer-to-peer international remittance platform, threat modeling identifies critical risks such as fraudulent transaction injection, man-in-the-middle attacks on transaction data, and unauthorized access to user account balances. Design mitigations include end-to-end encryption for transaction data, multi-factor authentication for all transactions above a certain threshold, robust input validation to prevent injection attacks, and anomaly detection for transaction patterns.
- **Secure Architecture Design**: Application and system architectures must be designed based on established secure design principles, including defense-in-depth, principle of least privilege, secure defaults, minimizing attack surface, and clear separation of duties. For financial applications, this includes secure design of transaction processing flows and data storage.
- **Data Flow Diagrams (DFDs)**: DFDs are created and maintained to visualize how sensitive data (especially financial data and PII) is captured, processed, transmitted, stored, and destroyed by the application. DFDs help identify potential points of data exposure or leakage.
- **Secure System Engineering Principles**: Development must adhere to secure system engineering principles as outlined in frameworks like ISO 27001 Annex A.14.2.5, ensuring security is an integral part of the engineering process. This includes considerations for secure component selection and integration.

- **API Security Design:** For applications exposing or consuming APIs, a security-first design approach for APIs is mandatory, including robust authentication, authorization, input validation, rate limiting, and encryption, aligning with OWASP API Security Top 10.

3. Implementation (Coding) Phase:

- **Adherence to Secure Coding Standards:** All code must be developed in accordance with World Bank's Secure Coding Standards, which are based on industry best practices such as the OWASP Top 10 Web Application Security Risks and the SANS/CWE Top 25 Most Dangerous Software Errors. Key practices include, but are not limited to:
 - **Input Validation:** Rigorous validation of all input data (from users, other systems, files) to prevent injection flaws (SQLi, XSS, Command Injection), buffer overflows, and other input-based attacks.
 - **Output Encoding:** Proper encoding of output data to prevent XSS and other content-injection attacks when rendering data in user interfaces or other systems.
 - **Authentication and Session Management:** Implementation of strong, secure authentication mechanisms and robust session management (e.g., secure session tokens, timely session expiration, protection against session fixation).
 - **Access Control:** Enforcement of authorization checks at each point of access to functions and data, based on the principle of least privilege.
 - **Cryptography:** Correct use of Bank-approved cryptographic libraries and algorithms for encryption of sensitive data at rest and in transit, and for secure hashing of credentials. Avoidance of custom cryptography.
 - **Error Handling and Logging:** Secure error handling that does not reveal sensitive system information. Comprehensive logging of security-relevant events (see Logging and Monitoring Policy).
 - **Secure File Handling:** Secure practices for file uploads, downloads, and processing to prevent path traversal and malware introduction.
- **Use of Approved Libraries and Components:** Only approved, vetted, and up-to-date third-party libraries, frameworks, and open-source components may be used. Software Composition Analysis (SCA) tools are employed to identify and manage vulnerabilities in these components. A Bill of Materials (BOM) for software components is maintained.

- **Secrets Management:** All secrets (passwords, API keys, encryption keys, certificates) must be managed through World Bank's approved secrets management solution. Secrets must never be hardcoded in source code, configuration files, or build scripts.
- **Peer Code Reviews:** Security-focused peer code reviews are mandatory for critical modules and security-sensitive code changes.
- *Security Manager's Perspective:* World Bank mandates the integration of Static Application Security Testing (SAST) tools directly into the Integrated Development Environment (IDE) and the Continuous Integration/Continuous Deployment (CI/CD) pipeline. Builds automatically fail if new critical or high-severity vulnerabilities are introduced, enforcing a "secure by default" coding culture.

4. Testing Phase:

- **Static Application Security Testing (SAST):** Automated SAST scans are integrated into the CI/CD pipeline and performed regularly on the codebase to identify vulnerabilities like those listed in OWASP Top 10 and CWE Top 25 during the development process.
- **Dynamic Application Security Testing (DAST):** DAST tools are used to test running applications in staging or UAT environments to identify runtime vulnerabilities, such as XSS, SQL injection, and insecure configurations.
- **Software Composition Analysis (SCA):** SCA tools are used to scan for known vulnerabilities in third-party and open-source components used by the application.
- **Manual Security Code Reviews:** For high-risk applications or critical components (e.g., authentication modules, payment processing logic), manual security code reviews are conducted by the Cybersecurity Team or qualified third parties.
- **Penetration Testing:** Independent penetration testing (internal or external) is mandatory for all critical and high-risk applications (especially those handling financial data or exposed to the internet) prior to initial deployment and periodically thereafter (e.g., annually or after major changes).
- **API Security Testing:** Specific security testing for APIs, focusing on authentication, authorization, input validation, and protection against common API attacks.
- **Business Logic Flaw Testing:** Testing for flaws in the application's business logic that could be exploited for fraudulent activities, particularly in financial transaction processing (e.g., testing for race conditions, unauthorized

transaction modifications, ensuring proper segregation of duties in transaction approvals).

- **Fuzz Testing:** Employing fuzz testing techniques for critical input vectors to discover unexpected behavior and potential vulnerabilities.
- **Use of Production-Like Test Data:** When testing, production data must not be used in non-production environments unless it has been appropriately anonymized or tokenized and the test environment meets the same security standards as production.
- *Use Case (Security Testing for Mobile Banking App):* Before launching a new version of the World Bank mobile banking application, a comprehensive security testing phase is executed. This includes:

1. SAST scans on the mobile application code (iOS and Android).
2. DAST scans on the backend APIs supporting the mobile app.
3. SCA to check for vulnerabilities in third-party SDKs used.
4. A specialized mobile application penetration test conducted by an external firm, focusing on issues like insecure data storage on the device, insecure communication, and platform-specific vulnerabilities. The penetration test identifies a vulnerability where sensitive transaction details are cached insecurely on the device. This vulnerability is remediated and re-tested before the app version is released.

5. Deployment Phase:

- **Secure Configuration:** Applications and their underlying infrastructure (servers, databases, web servers) must be deployed with secure configurations, adhering to World Bank's Secure Configuration Baselines (see Technological Controls section). This includes removing default credentials, disabling unnecessary services, and applying security hardening.
- **Secure Deployment Pipeline (CI/CD Security):** Deployment processes must be automated and secured. This includes securing the CI/CD tools themselves, managing credentials used in the pipeline securely, and verifying the integrity of deployment artifacts.
- **Change Management:** All deployments to production environments must follow the formal Change Management Policy.
- **Post-Deployment Verification:** After deployment, security controls and application functionality must be verified in the production environment to ensure the deployment was successful and no new issues were introduced.

6. Maintenance Phase:

- **Continuous Vulnerability Monitoring:** Applications in production must be continuously monitored for new vulnerabilities (see Vulnerability Management Procedure). This includes regular rescanning and monitoring of threat intelligence feeds.
- **Security Patching:** Timely application of security patches for the application itself and all its underlying components (OS, libraries, frameworks) is mandatory (see Patch Management Policy).
- **Periodic Security Re-assessment:** Critical applications must undergo periodic security re-assessments, including penetration tests, especially after significant changes or on an annual basis.
- **Secure Decommissioning:** When an application is retired, it must be decommissioned securely, ensuring all sensitive data is appropriately archived or destroyed according to the Data Retention Policy and Asset Lifecycle Document. Access to the application and its components must be revoked.

Developer Training and Awareness

World Bank is committed to ensuring that all personnel involved in software development possess the necessary knowledge and skills to build secure applications.

- **Mandatory Secure Coding Training:** All developers (employees and contractors) must complete mandatory secure coding training upon onboarding and annually thereafter. This training covers common vulnerabilities (OWASP Top 10, SANS CWE Top 25), secure coding principles, and World Bank specific security standards.
- **Role-Specific Training:** Specialized security training is provided for roles dealing with particularly sensitive areas, such as API development, mobile application development, or cloud-native development.
- **Threat Awareness Updates:** Regular updates and briefings are provided to development teams on emerging threats, new vulnerability types, and evolving attacker techniques relevant to financial applications.

Framework Alignment

These Secure Development Guidelines are aligned with and draw from leading industry standards and frameworks, including:

- **NIST Special Publication 800-218 (Secure Software Development Framework - SSDF):** This guideline incorporates the practices and tasks outlined in the SSDF to integrate security throughout the SDLC.

- **ISO 27001/ISO 27002:** Specifically, controls related to security in development and support processes (e.g., Annex A.14.2 in ISO 27001:2013, which maps to controls like A.8.25, A.8.26, A.8.28, A.8.32 in ISO 27001:2022).
- **OWASP (Open Web Application Security Project):** Guidance such as the OWASP Top 10, OWASP ASVS (Application Security Verification Standard), and OWASP Secure Coding Practices Guide are key references.
- **SANS/CWE:** The SANS/CWE Top 25 Most Dangerous Software Errors list informs our focus on preventing common coding errors that lead to vulnerabilities.
- **FFIEC Development and Acquisition Guidelines:** Relevant FFIEC guidance for financial institutions regarding software development and acquisition is incorporated.

Real-World Example: Development of an Internal Financial Reporting Tool

A World Bank development team is tasked with creating a new internal tool for generating complex financial reports for senior management.

- **Requirements:** The tool will process highly sensitive, non-public financial data. Key security requirements include: strong role-based access control to limit report generation and viewing to authorized personnel, end-to-end encryption of data, comprehensive audit trails of report generation and access, and protection against data leakage.
- **Design:** Threat modeling focuses on insider threats (unauthorized access by employees) and data exfiltration risks. The architecture incorporates a multi-tier design with strict separation between data processing, business logic, and presentation layers. All sensitive computations are performed on a secure backend server.
- **Implementation:** Developers follow World Bank's secure coding standards, paying particular attention to input validation for report parameters (to prevent injection that could alter queries) and ensuring all database queries use parameterized statements. Sensitive data within the database is encrypted at the field level using Bank-approved encryption libraries. All access to the tool and report generation activities are logged with detailed user context.
- **Testing:** SAST tools are integrated into the CI/CD pipeline. DAST is performed on the staging environment. The Cybersecurity Team conducts a manual code review of the authentication, authorization, and data encryption modules. A targeted penetration test is performed to simulate insider attempts to bypass access controls or exfiltrate data.

- **Deployment:** The tool is deployed on hardened servers within a secure internal network segment. Access is restricted via an IAM group linked to specific job roles defined by Finance management.
- **Maintenance:** The tool is included in the Bank's regular vulnerability scanning schedule. Any identified vulnerabilities are addressed according to the Vulnerability Management Procedure.

Table: SSDLC Security Activities and Responsibilities

SDLC Phase	Security Activity	Responsible (Primary)	Supporting Roles	Tools/Techniques Example(s)	Output/Artifact
Requirements	Define Security Requirements & Data Classification	Product Owner	Security, Dev, Legal	Workshops, Policy Review, DPIA	Security Requirements Document, Data Classification
	Identify Compliance Needs	Compliance/Legal	Product Owner, Security	Regulatory Matrix Analysis	Compliance Requirements List
Design	Conduct Threat Modeling	Security Team	Dev Team, Architects	STRIDE, DREAD, Process Flow Diagrams	Threat Model Document, Identified Mitigations
	Secure Architecture Review	Architects	Security Team	Design Patterns, Security Principles	Secure Architecture Document
	Create Data Flow Diagrams	Dev Team/Architects	Security Team	DFD Tools (e.g., Visio, Lucidchart)	Data Flow Diagrams
Implementation	Adhere to Secure Coding Standards	Development Team	Security Champions	OWASP Top 10, SANS CWE Top 25, Bank Coding Standards	Secure Code
	Use Approved Libraries & SCA	Development Team	Security Team	SCA Tools (e.g., OWASP Dependency-Check, Snyk)	Software Bill of Materials (SBOM), Vulnerability List
	Implement Secure Secrets Management	Development Team	Ops, Security	HashiCorp Vault, Azure Key Vault	No Hardcoded Secrets
	Conduct Security-focused Peer Code Reviews	Development Team	Security Champions	Manual Review Checklist	Code Review Records
Testing	Static Application Security Testing (SAST)	Development Team	Security Team	SAST Tools (e.g., Checkmarx, Veracode, SonarQube)	SAST Reports, Remediated Vulnerabilities
	Dynamic Application Security Testing (DAST)	QA/Security Team	Development Team	DAST Tools (e.g., OWASP ZAP, Burp Suite Pro)	DAST Reports, Remediated Vulnerabilities

	Penetration Testing (for high-risk apps)	Security Team/3rd Party	Development Team	Pen Test Methodologies (PTES, OSSTMM)	Penetration Test Report, Remediated Vulnerabilities
	API Security Testing	QA/Security Team	Development Team	API Fuzzers, Postman Security Tests	API Test Reports
	Business Logic Flow Testing	QA/Business Analysts	Development Team	Use Case Analysis, Scenario Testing	Logic Flow Test Report
Deployment	Secure Configuration of Environments	Operations Team	Security Team	CIS Benchmarks, Bank Baselines, IaC Security Scanners	Hardened Environment
	Secure CI/CD Pipeline	DevOps/Ops Team	Security Team	Pipeline Security Tools (e.g., Jenkins Security Plugins)	Secure Deployment Process
	Post-Deployment Security Verification	Operations/QA Team	Security Team	Smoke Tests, Configuration Audits	Deployment Verification Report
Maintenance	Continuous Vulnerability Monitoring	Security Operations	App/System Owners	Vulnerability Scanners, Threat Intelligence Feeds	Ongoing Vulnerability Reports
	Timely Security Patching	App/System Owners	IT Operations	Patch Management System	Patch Compliance Records
	Periodic Security Re-assessment	Security Team	App/System Owners	SAST, DAST, Pen Testing	Updated Risk Assessments

- Value of Table:** This table provides a clear, structured, and actionable matrix detailing specific security activities, primary responsibilities, supporting roles, example tools/techniques, and expected outputs for each phase of the SDLC. For a complex organization like World Bank, with numerous development projects and stringent regulatory demands (e.g., FFIEC guidelines on development and acquisition), this matrix is invaluable. It promotes accountability by clearly assigning responsibilities, ensures comprehensive security coverage throughout the lifecycle (supporting "shift-left" principles), facilitates easier auditing by providing a clear map of security integration, and helps in standardizing secure development practices across diverse development teams, whether internal or external. This ultimately leads to more resilient and secure financial applications.