Foxit Development Security Practices

Table of Contents

Foxit Development Security Practices:

Foxit Software is committed to building secure, reliable, and trustworthy products through a mature and continuously improving Secure Software Development Lifecycle (S-SDLC). Security and privacy are integrated into every stage of product development and operations, rather than treated as afterthoughts. This comprehensive approach helps reduce security vulnerabilities, minimize the impact of security incidents, and shorten remediation timelines, thereby protecting customer data and ensuring business continuity.

Foxit's Development Security practices are aligned with industry-recognized standards and frameworks, informed by Microsoft's Security Development Lifecycle (SDL), while being adapted to Foxit's specific products, technologies, and agile development methodologies.

Security Objectives

Foxit's Development Security program is designed to achieve three core objectives:

- **Reduce security issues** through secure design principles, secure coding practices, and proactive security testing throughout the development lifecycle.
- **Reduce the impact of security issues** by implementing layered security controls, establishing resilience planning, and maintaining disaster recovery capabilities.
- **Shorten the time to remediation** by operating a structured security issue response process and integrating security activities into continuous development workflows.

Strategic Approach

Foxit's security strategy emphasizes proactive, organization-wide practices to embed security throughout the entire product lifecycle:

- **Shift-Left Security**: Security considerations are integrated from the earliest stages of product development, including requirements analysis, architecture design, and initial implementation. Early attention to security helps prevent vulnerabilities, reduce risk exposure, and streamline remediation efforts.
- **Culture and Education**: Security responsibilities are clearly defined across all roles and reinforced through comprehensive, role-based training programs that build security awareness and competence.
- **Full Lifecycle Security**: Security activities span the entire software lifecycle, from initial design and implementation through verification, release, and post-deployment monitoring. This comprehensive approach provides end-to-end traceability, clear accountability, and consistent security oversight throughout the product's lifecycle.

Standards and Best Practices

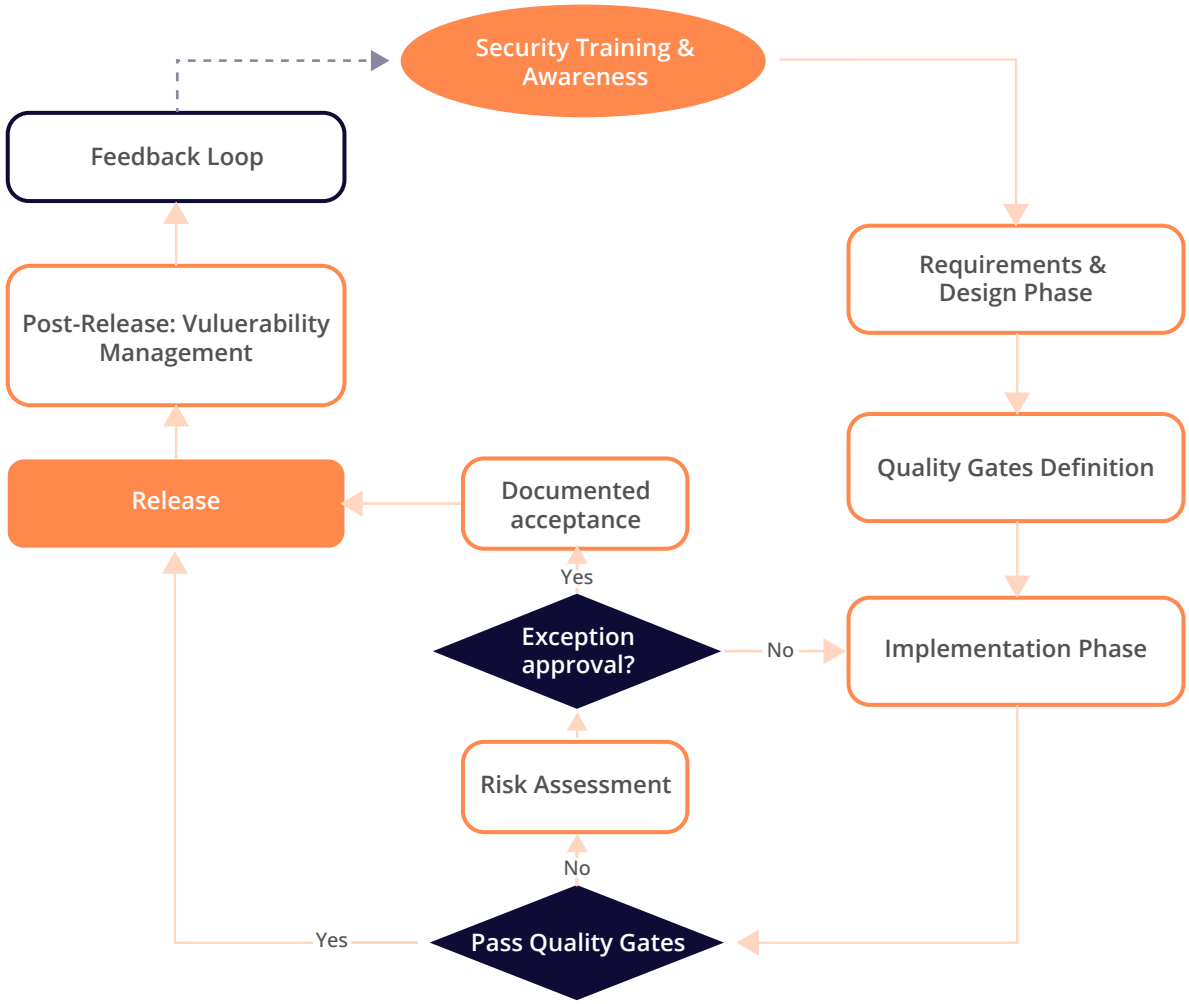
Foxit references widely adopted industry standards and security frameworks to guide secure design, implementation, validation, and operations. These include, but are not limited to:

- OWASP Top 10 and OWASP Application Security Verification Standard (ASVS)
- OWASP Web Security Testing Guide (WSTG)
- OWASP Software Assurance Maturity Model (SAMM)
- OWASP privacy and secure coding guidance
- Industry secure coding guidelines and cryptographic best practices

Compliance and regulatory requirements are incorporated into security requirements during the product requirements and design phases to ensure products meet applicable legal and industry obligations from the outset.

Secure Software Development Lifecycle (S-SDLC)

Foxit's S-SDLC defines mandatory security activities across the full product lifecycle. These activities are enforced through defined processes, quality gates, and collaboration between development teams and the Product Security Team.



Training and Awareness

Security is a shared responsibility at Foxit. All employees complete mandatory security awareness training, with specialized role-specific training provided to developers, architects, and project leaders. Training modules cover secure development fundamentals, common vulnerability patterns, threat modeling concepts, and evolving security risks.

Objective: Ensure that all personnel understand their secure development responsibilities and can apply security best practices in their daily work.

Controls include:

- Mandatory Security training for all employees, with role-based modules tailored to developers, architects, and project leaders.
- Periodic refresh training addressing emerging threats, secure design practices, and lessons learned from prior security incidents.

Foxit tracks the completion rates of mandatory and role-specific secure development training, along with the frequency of refresher sessions addressing emerging threats and secure design practices. These metrics ensure that all team members maintain the necessary knowledge and awareness to implement secure coding and design principles effectively.

Requirements and Design

Security and privacy requirements are defined at the outset of each project and continuously updated throughout the product lifecycle. These requirements consider data sensitivity, evolving threat landscapes, regulatory obligations, and lessons learned from previous security incidents.

Threat modeling is a mandatory design activity. Development teams systematically analyze system architecture, data flows, and trust boundaries to identify and prioritize potential threats. Identified risks are formally documented, tracked, and addressed through defined mitigation measures before release.

Objective: Ensure security and privacy are integrated from the earliest stages of design and maintained throughout the product lifecycle.

Controls include:

- Formal software security requirement analysis for new projects and significant feature changes.
- Threat modeling and attack surface analysis during the design phase, with automated and manual support where applicable.
- Continuous maintenance and review of security requirements, including version control and traceability to design and implementation artifacts.
- Documentation and tracking of identified risks and corresponding mitigation measures.

Metrics in this stage focus on the proportion of projects with formal security requirement analyses and threat modeling completed, as well as the number of identified risks that are tracked and mitigated prior to release. These measurements provide visibility into the effectiveness of early-stage security planning and risk management.

Quality Gates

Foxit establishes minimum acceptable security quality thresholds that all products must meet before release. These thresholds define severity-based remediation requirements, ensuring that high-risk and critical vulnerabilities are addressed within predefined timeframes. Quality gates are periodically reviewed and may be adjusted to reflect product-specific risk profiles while maintaining consistent security expectations company-wide.

Objective: Enforce minimum security quality standards and ensure vulnerabilities are appropriately mitigated prior to release.

Controls include:

- Defined vulnerability severity classifications aligned with internal and industry standards
- Mandatory remediation of critical and high-risk vulnerabilities before release
- Risk-based acceptance process for exceptional cases, with documented approvals and justifications
- Periodic review and adjustment of quality gate criteria based on evolving threats and product risk profiles

During implementation, Foxit monitors the coverage of automated security scans per release cycle, the number of high and critical vulnerabilities detected and remediated before release, and the proportion of third-party components reviewed and approved based on their security posture. These indicators ensure that secure coding practices and third-party governance are effectively applied.

Implementation

During implementation, Foxit enforces security controls and coding practices to prevent vulnerabilities from entering production.

Controls include:

- Defined secure coding standards and guidelines applicable to all development teams
- Integration of automated security checks within development workflows to identify potential security issues early
- Enforcement of secure configuration baselines and build-time hardening requirements
- Governance over third-party and open-source components based on security risk assessment and approval processes
- Processes to detect, track, and remediate exposed credentials or sensitive information

Metrics include coverage of automated security scans, remediation of high/critical issues, and proportion of third-party components reviewed.

Verification & Release

All products undergo a structured security verification process prior to release to confirm compliance with Foxit's security baseline and defined quality standards. Release approval is contingent on the remediation of identified security risks or their formal acceptance in accordance with company policy. This phase focuses on validating that implemented security controls function as intended in integrated and operational environments.

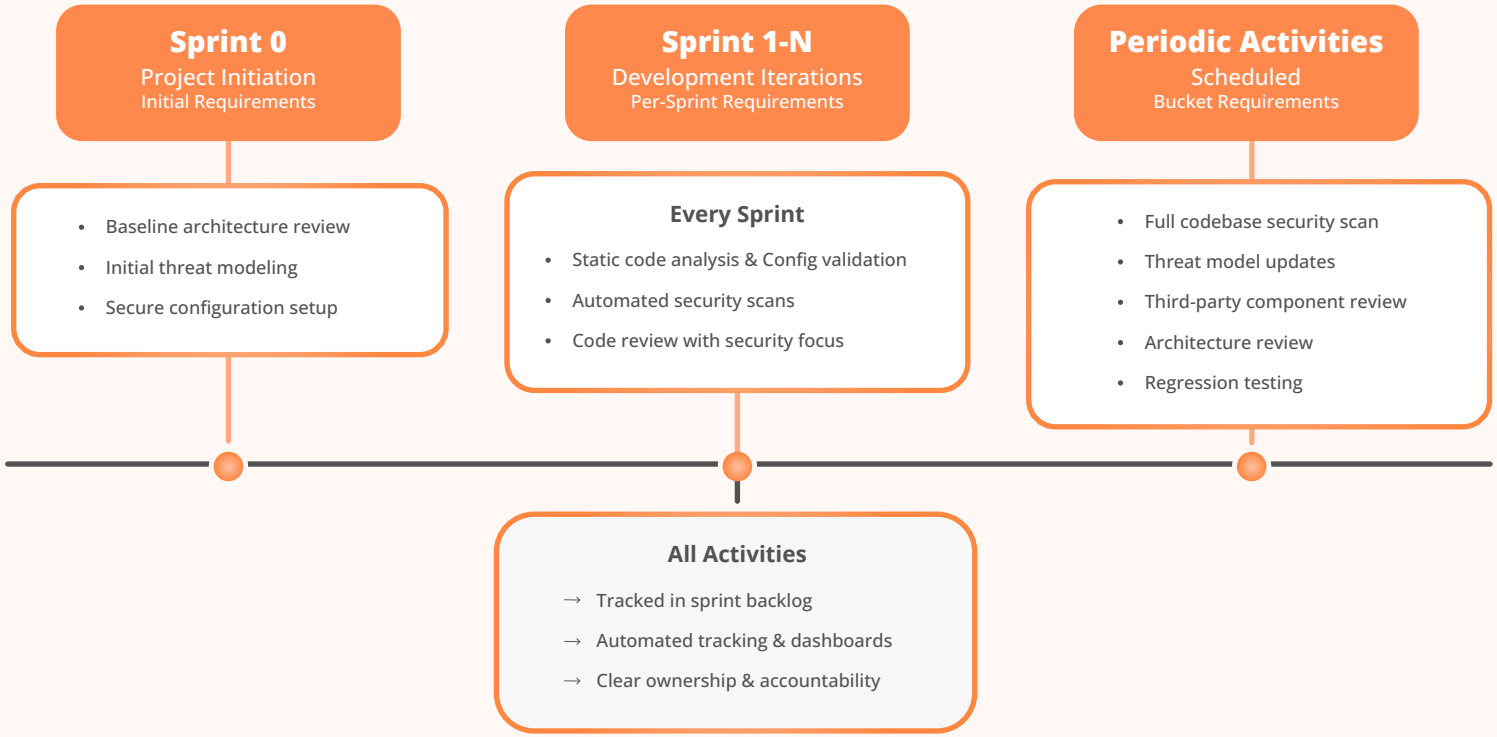
Controls include:

- Pre-release security validation activities to assess the effectiveness of implemented security controls in integrated and operational environments
- Security testing of externally exposed services and critical components to identify implementation, configuration, and robustness weaknesses
- Security regression verification to confirm that previously identified vulnerabilities have been effectively remediated and do not recur
- A final, cross-functional security review involving Security, QA, and Product teams to confirm compliance with Foxit's security baseline and quality standards
- A formal release approval process enforcing defined security quality gates, including documented remediation or risk acceptance and cross-departmental sign-off

Metrics include coverage of pre-release security validation and remediation timeliness.

SSDLC Agile Integration

Security tasks defined in SSDLC are incorporated into each sprint and backlog to maintain consistent security practices.



Security Activity Categorization

Security activities are organized based on frequency and criticality:

- **Initial Requirements**: Tasks essential at project initiation, such as baseline architecture review, initial threat modeling, and secure configuration setup.
- **Per-Sprint Requirements**: Tasks executed in every iteration, e.g., static code analysis, configuration validation, and automated security scans on active code changes.
- **Periodic (Bucket) Requirements**: Recurring tasks on a scheduled basis, including full codebase scans, updates to threat models, review of third-party components, annual architecture reviews, and regression testing to verify previously remediated issues.

Implementation Approach

Security tasks are incorporated into sprint planning and backlog management, with assignments to developers, security champions, and reviewers. Automated tracking, dashboards, and documentation maintain traceability and accountability throughout the agile process. Security activities are periodically reviewed to ensure alignment with evolving threats, regulatory obligations, and project-specific risk profiles.

Layered Security Activities

Foxit conducts a comprehensive set of **security testing and verification activities** within the SSDLC. These activities are supported by automated and manual tools to ensure early detection of vulnerabilities, effective tracking of security issues, and robust governance over security artifacts.

Testing

Foxit employs a layered approach to security testing, combining automated and manual techniques to detect vulnerabilities and validate security controls. Testing tools include, but are not limited to, the following types:

Code Quality & Static Analysis

- **Static Code Analysis**: Linting tools, IDE-integrated compile-time warnings, and static analysis plugins provide immediate feedback during development and are integrated into CI/CD workflows. Enterprise-grade solutions, perform scheduled full-codebase analysis to identify complex vulnerabilities that may not be captured by lightweight tools.
- **Credential and Secret Scanning**: Detect potential disclosure of credentials or sensitive information in source code and configuration files.
- **Encryption Scanning**: Verify that encryption practices in code and execution follow security best practices.

Runtime & Dynamic Testing

- **Fuzz Testing**: Test APIs and system components with malformed or unexpected input to uncover vulnerabilities and validate error handling.
- **Dynamic Application Security Testing (DAST)**: Perform automated runtime testing of web applications to detect vulnerabilities such as injection flaws, authentication issues, and misconfigurations.

- **Penetration Testing**: Conduct targeted attacks on systems prior to release to validate security controls and ensure that security requirements are correctly implemented.

Configuration & Component Governance

- **Configuration Verification**: Assess production system configurations against security standards and best practices.
- **Component Governance**: Inspect open-source and third-party components for versions, vulnerabilities, and licensing compliance to prevent known risks from entering products.
- **Docker Image Scanning**: Analyze container images for known vulnerabilities and misconfigurations to maintain secure deployment.

Depending on the focus and efficiency, these testing activities are carried out using **automated tools, manual methods, or a combination of both**, and may be integrated into CI/CD workflows or executed on a scheduled (periodic) basis by security experts.

Management

To ensure traceability, governance, and accountability across all security activities, Foxit carries out the following management activities:

Centralized Defect Management

Track, prioritize, and resolve security defects across products, ensuring consistent handling and visibility of findings. Supported by centralized defect management platforms.

Security Task Management

Manage security-related tasks and requirements, including threat modeling, remediation actions, security reviews, and risk acceptance, with clear ownership, status tracking, and accountability. Supported by task and work item tracking systems.

Documentation & Visualization

Store and maintain security requirements, design decisions, threat models, policies, and test results to ensure traceability, audit readiness, and institutional knowledge retention. Visualize system and application architectures to understand dependencies, security boundaries, and potential risk areas during design and review. These activities are supported by documentation, knowledge management, and architecture visualization tools.

These management activities are integrated into standard SDLC workflows, embedding security tasks, defects, and approvals into day-to-day development processes. This ensures **end-to-end visibility, ownership, and auditability** throughout the software lifecycle.

Testing and management activities operate together within the SSDLC to support structured security practices. Testing activities focus on identifying potential vulnerabilities and validating security controls, while management activities capture and organize findings, decisions, and remediation actions to ensure traceability and accountability. This combination provides end-to-end visibility into security practices throughout the development to lifecycle, enabling clear tracking of issues, actions taken, and process outcomes.

Vulnerability Management

Foxit Software maintains a structured Vulnerability Management program to identify, assess, remediate, and communicate security issues across its products and services. This program ensures that vulnerabilities are addressed in a timely, transparent, and accountable manner, reducing risk for customers and enhancing the overall security posture of Foxit products.

Key Objectives

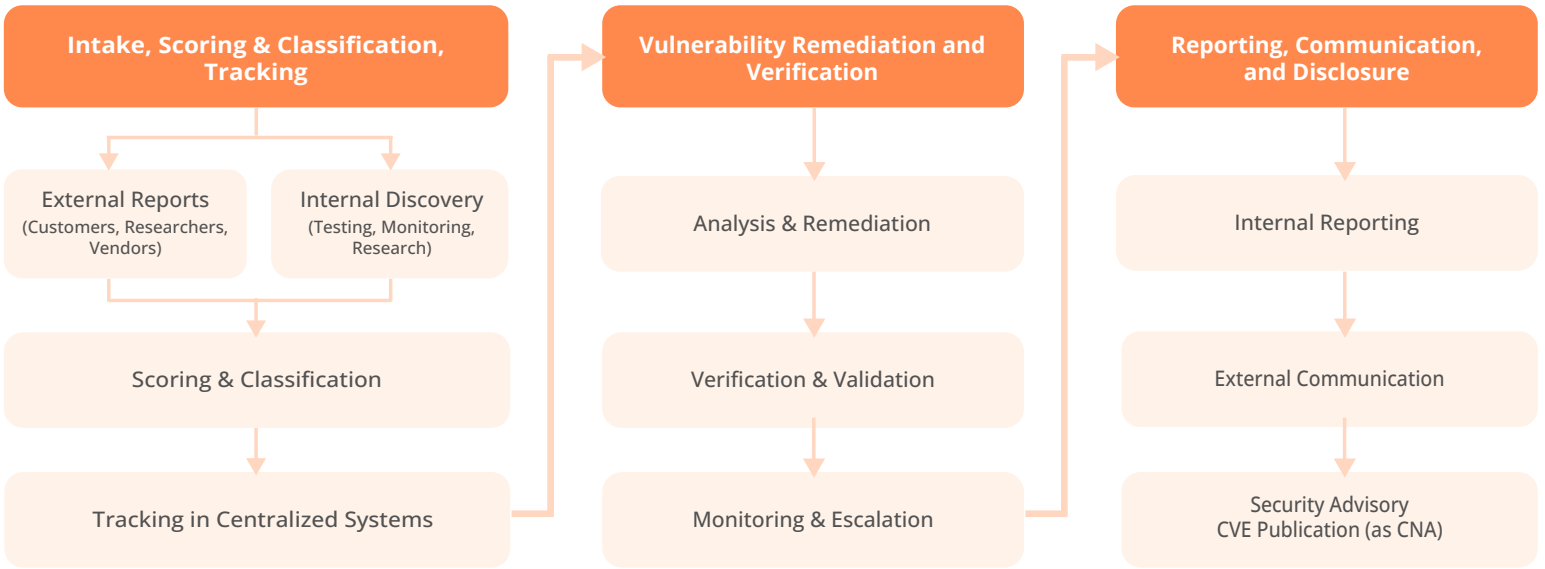
- **Timely Identification**: Capture vulnerabilities reported externally or discovered internally through security testing, monitoring, or research.
- **Risk Assessment**: Classify vulnerabilities based on severity using industry-standard frameworks such as CVSS v3.0 to guide prioritization and remediation timelines.
- **Effective Remediation**: Assign vulnerabilities to the appropriate development teams, track remediation, and verify fixes through testing.
- **Transparent Communication**: Provide stakeholders with structured notifications and, where appropriate, public advisories.

CVE Numbering Authority (CNA) Role

Foxit Software is a CVE Numbering Authority (CNA), authorizing us to assign CVE identifiers for newly discovered vulnerabilities affecting our products. As a CNA, Foxit ensures all vulnerabilities are tracked and publicly referenced in a standardized and transparent manner, enabling customers and security researchers to correlate vulnerabilities with CVE IDs and build confidence in our remediation and disclosure process.

Response Process

Foxit's Vulnerability Management process ensures timely identification, assessment, remediation, and communication of security vulnerabilities, providing accountability and transparency to protect customers and strengthen product security.



Intake, Scoring & Classification, Tracking

- **Intake**: Vulnerabilities may originate from external reports (e.g., customers, security researchers, vendors) or internal discovery (e.g., penetration testing, static/dynamic analysis, fuzzing, component review).
- **Scoring & Classification**: Severity is assessed using CVSS v3.0, mapped to a level to guide prioritization and remediation timelines.
- **Tracking**: All vulnerabilities are recorded and tracked to ensure ownership, accountability, and traceability throughout the remediation process.

Remediation and Verification

- **Analysis & Remediation**: Vulnerabilities are assigned to responsible development teams for remediation according to severity and impact.
- **Verification & Validation**: Remediation is verified and validated through testing, including regression testing, static/dynamic analysis, and penetration testing, to confirm vulnerabilities are effectively addressed.
- **Monitoring & Escalation**: For critical vulnerabilities, timelines and progress are closely monitored, with escalation procedures followed as needed.

Reporting, Communication, and Disclosure

- **Internal Reporting**: Security Committee, Product Security Team, and development teams are regularly updated on vulnerability status and remediation progress.
- **External Communication**: Customers and security researchers receive acknowledgments, status updates, and post-remediation notifications.
- **Disclosure**: Once a vulnerability is resolved, CVE identifiers are published to provide a standardized reference for customers and the broader security community.

Metrics and Oversight

Foxit monitors key indicators such as the number of vulnerabilities reported, time-to-remediation, CVE assignments, and verification success rates. These metrics help ensure the Vulnerability Management program remains effective, accountable, and aligned with Foxit's security objectives.

Conclusion

Foxit's Development Security program integrates security as a core strategic enabler across the entire software lifecycle. Through structured processes, clearly defined responsibilities, and continuous validation, security is not just a technical control—it directly supports business objectives, regulatory compliance, and customer trust. By aligning with industry best practices and proactively embedding security from design to deployment, Foxit ensures that its products not only protect customer data but also strengthen organizational resilience and sustain long-term strategic value.

By continuously refining its S-SDLC and associated security practices, Foxit ensures that security remains an integral and evolving part of product development, enabling the organization to adapt to emerging threats, regulatory changes, and evolving customer expectations.


Foxit