



Secure Coding Standard

DOC-IT-IS-33
Revision 1.0
NOVEMBER 1, 2025

REVISION HISTORY

This revision history lists all changes to this document, the date of each revision, and the reason for each revision.

<u>Rev.</u>	<u>Rev. Date</u>	<u>Revision Description</u>
1.0	11/01/2025	Initial Version

Table of Contents

1 Executive Summary	4
2 Purpose.....	4
3 Scope.....	4
4 Secure Coding Standard.....	5
4.1 General Secure Coding Expectations	5
4.2 Input Validation.....	5
4.3 Output Encoding.....	5
4.4 Authentication & Authorization	6
4.5 Database & Storage Security	6
4.6 Secret Management.....	6
4.7 Cryptography.....	6
4.8 Dependency & Supply Chain Security	7
4.9 Error Handling & Logging	7
4.10 Secure Framework and Language Usage.....	7
4.11 Serialization & Deserialization.....	7
4.12 Session Management & Cookies	7
4.13 Secure System Design.....	8
4.14 API Security.....	8
4.15 File & Data Handling	8
4.16 Secure Build & Release Practices.....	8
4.17 Client-Side and JavaScript Security	8
4.18 Deployment & Infrastructure.....	9
4.19 Monitoring, Logging, and Alerting.....	9
4.20 Verification & Testing	9
4.21 Documentation Expectations.....	9
5 Roles & Responsibilities	10
5.1 Application Owners	10
5.2 Information Security Team	10
6 Compliance	10
7 Exceptions	10
8 Review & Updates	10
9 References.....	10

1 Executive Summary

This document provides guidelines to help developers write software that is resistant to security vulnerabilities. These standards protect against common threats and ensure the security of applications.

2 Purpose

The purpose of this Secure Coding Standard is to ensure that all internally developed software is designed, implemented, and maintained using secure development practices. This standard defines expected behaviors, minimum security requirements, and best-practice recommendations for all developers and system designers.

3 Scope

This standard applies to RXO, Inc., including all subsidiaries, divisions, and operating entities. All directors, officers, full-time and part-time employees, agents, contractors, and third parties acting on behalf of RXO must comply with this policy.

This standard applies to:

- All custom-developed applications.
- All modifications to existing software.
- All CI/CD pipelines, application components, libraries, and dependencies.
- Web, mobile, desktop, cloud-native, and server-side applications.
- APIs, microservices, services, and backend log.

4 Secure Coding Standard

4.1 General Secure Coding Expectations

Assume external systems and input cannot be trusted.

- All inputs, requests, and integrations should be validated before use.

Do not attempt to repair malformed input.

- Reject or fail the request when input does not meet expectations.
- Use approved patterns, frameworks, libraries, and pipeline templates whenever available.

Design systems with failure in mind.

- Fail closed, not open. Ensure partial transactions are rolled back and not resumed mid-flow.

Follow applicable laws, regulations, and internal compliance requirements.

- Seek clarification from InfoSec or Legal when needed.

Threat model important systems and features.

- Document findings and mitigate meaningful threats.

Keep frameworks, libraries, and dependencies up to date.

- Address dependency security issues as part of normal development work.

Retest functionality after bug fixes or feature changes.

- Security regression testing should accompany any material system change.

Use secure development tooling.

- SAST, SCA, secret scanning, and linting tools should run on every merge or commit, where applicable.

4.2 Input Validation

- Validate all input from users, APIs, files, and external systems.
- Prefer allowlists over blocklists.
- Escape or sanitize input where appropriate for the output context.
- Validate file uploads (type, size, extension, content).
- Avoid inline code execution based on user input.

4.3 Output Encoding

- Encode all user-controlled output according to its context (HTML, JavaScript, URL, JSON, etc.).

- Treat encoding as a last step before rendering or transmitting data.
- Avoid inline JavaScript or other inline execution where possible.

4.4 Authentication & Authorization

- Use platform-provided authentication and session-management components whenever possible.
- Do not implement custom authentication or session logic unless formally approved.
- Validate authorization on every request, including API requests.
- Protect all sensitive operations using least-privilege principles and role-based access controls.

4.5 Database & Storage Security

- Always use parameterized queries for all database operations.
- Prefer ORM frameworks where available.
- Avoid dynamic SQL and string concatenation for building queries.
- Protect database servers, host systems, and network segments through hardening and access control.

4.6 Secret Management

- Never store secrets in code, configuration files, or version control.
- Use approved secret-management tools for all sensitive values.
- Use secret-scanning tools on all repositories and pipelines.
- Rotate exposed secrets immediately.

4.7 Cryptography

- Follow the RXO Cryptography Policy
- All network communication must occur over HTTPS or approved encrypted transports.
- Use only approved cryptographic algorithms and key sizes.
- Follow internal cryptographic standards or industry best practices (e.g., NIST).
- Store all sensitive data encrypted in transit and at rest, where applicable.

4.8 Dependency & Supply Chain Security

- Use approved and trusted package repositories.
- Monitor dependencies for security vulnerabilities.
- Remove outdated, unused, or unsupported components.
- Ensure build systems, CI/CD, and source control platforms are securely configured.
- Lock down developer environments and IDEs as part of supply chain control.

4.9 Error Handling & Logging

- Handle errors gracefully, without exposing sensitive information.
- Log errors in a central logging system.
- Implement monitoring and alerting for security-relevant events.
- Avoid displaying stack traces or internal details to end users.

4.10 Secure Framework and Language Usage

- Use the latest stable version of any programming framework.
- Avoid deprecated or unsupported features.
- Prefer memory-safe languages when feasible.
- If using languages without automatic memory safety, apply strong input validation and safe function usage.

4.11 Serialization & Deserialization

- Do not deserialize objects from untrusted sources.
- If unavoidable, implement strict validation and security controls.
- Avoid insecure deserialization patterns across all languages.

4.12 Session Management & Cookies

- Use secure cookie attributes (Secure, HttpOnly, SameSite) where appropriate.
- Manage session lifecycles securely.
- Avoid storing sensitive data in browser storage when possible.

4.13 Secure System Design

- Apply least privilege, defense in depth, attack surface reduction, and zero trust principles.
- Design for usability to encourage secure usage and reduce workarounds.
- Limit backwards compatibility where it introduces risk.
- Threat model early and revisit for significant architectural changes.

4.14 API Security

- Follow the RXO API Security Standard
- Enforce authentication and authorization on every API request.
- Validate all request parameters, headers, and payloads.
- Use rate limiting and resource controls.
- Document and secure all data flows involving sensitive data.

4.15 File & Data Handling

- Secure source control repositories with proper access control and monitoring.
- Protect application files, uploaded files, and stored assets.
- Back up critical data securely and encrypt when appropriate.
- Avoid using production data in non-production environments unless properly masked.

4.16 Secure Build & Release Practices

- Follow the organization's Secure SDLC process.
- Integrate automated testing (unit, integration, security) into pipelines.
- Perform code reviews for all changes.
- Use immutable builds or integrity checks where feasible.
- Treat compiler warnings as requiring attention.

4.17 Client-Side and JavaScript Security

- Avoid inline JavaScript and dynamic code execution (eval, new Function, etc.).
- Perform all sensitive operations on the server.
- Ensure frameworks' built-in protections are not disabled.

4.18 Deployment & Infrastructure

- Validate hostnames and certificates for external system calls.
- Follow hardening guides for servers, containers, and cloud resources.
- Rotate default passwords immediately.
- Avoid running services as root or equivalent privileged users.

4.19 Monitoring, Logging, and Alerting

- Following the RXO Information Logging & Monitoring Standard
- Enable logging and monitoring for all applications and APIs.
- Alert on suspicious activities (authentication anomalies, rate limits, injection attempts).
- Provide logs to centralized SIEM or monitoring tools.

4.20 Verification & Testing

- Conduct regular security testing (SAST, DAST, SCA, secrets scanning).
- Perform penetration testing for high-value or high-risk systems.
- Retest after remediation to ensure issues are resolved.
- Validate that all controls and configurations meet internal standards.

4.21 Documentation Expectations

- Document security design decisions.
- Keep architecture diagrams, data flow maps, and dependency lists updated.
- Maintain coding patterns and reusable components in a shared repository.
- Share lessons learned to reduce recurrence of similar issues.

5 Roles & Responsibilities

5.1 Application Owners

- 5.1.1 Application owners must document their applications and submit them to the InfoSec team for review and approval prior to implementation.
- 5.1.2 Documentation should also be provided to the Incident Response Team to ensure they are aware of any new potential threat surfaces.
- 5.1.3 Application owners must notify the Network Management Team of any new APIs or API partners so that the API Gateway may be configured as needed to support the new APIs or new API partners.
- 5.1.4 Application owners must ensure logging information is sent to the central logging system to meet all requirements of the Audits & Logs section above and of the Company Information Logging & Monitoring Standard.

5.2 Information Security Team

- 5.2.1 The InfoSec team will conduct security assessments of any new or updated applications.

6 Compliance

Non-compliance with this policy may result in access restrictions to affected systems or disciplinary actions as per RXO's guidelines.

7 Exceptions

Any exceptions to this policy must be approved in writing by the RXO Governance Risk and Compliance (GRC) team and must include a valid business justification.

8 Review & Updates

This standard will be reviewed annually, or upon major infrastructure or threat landscape changes, or as required due to changes in business or technical requirements.

9 References

RXO Systems Acquisition, Development and Maintenance Policy
RXO Cryptography Policy
RXO API Security Standard
RXO Information Logging & Monitoring Standard
NIST SP 800-218, Secure Software Development
OWASP Application Security Verification Standard (ASVS)

Certificate of Review

Secure Coding Standard

December 18th 2025 10:53:53 am

Name	Job title	Stakeholder type	Action	Timestamp
Mark Gutman		Policy Owner	Submitted	Tuesday, November 25, 2025 1:17 PM EST
Faisal Khan		Policy Approver	Approved	Tuesday, December 2, 2025 11:26 AM EST
Mark Gutman		Policy Admin	Published	Thursday, December 18, 2025 10:53 AM EST