NIST SP 800-218 Compliance Product Guide

Mapping the JFrog Software Supply Chain Platform to the Secure Software Development Framework (SSDF) 1.1.

PREPARED FOR U.S. PUBLIC SECTOR

Federal Civilian · Defense · Intelligence · Critical Infrastructure · Federal Service Integrators

August 2026

Ver. 2.0

SECTION 01

Executive Summary

ABSTRACT

Government agencies must release software reliably, confidently, and on schedule to continually empower public servants and better serve citizens with modern apps and digital services. Simultaneously, they also need to ensure that software is secure and compliant to prevent cyber-attacks and meet IT transformation goals. As federal mandates shift from passive compliance to active risk elimination, agencies need a software supply chain platform that enforces governance at the binary level, treating the compiled artifact as the single source of truth for everything that runs in a federal environment.

PRODUCT SHOWCASE

JFrog provides an end-to-end Software Supply Chain Platform for automating, managing, securing, distributing, and monitoring your containers, binaries and artifacts, metadata, and configurations — everything that goes into software development — as they advance from source code to production. It unifies developers, operations, and security teams to safeguard the entire software supply chain in a holistic, self-hosted, cloud, multi-cloud, or hybrid platform. Additional information on the JFrog Platform can be found in Appendix A.

KEY CAPABILITIES

- Secured Software Development Framework (SSDF)
- Policy-as-Code Governance
- Software Development Lifecycle (SDLC)

TARGET AUDIENCE

- Federal Service Integrators (FSI)
- Government Direct — Federal Civilian, Defense, Intelligence, Critical Infrastructure, Other

SECTION 02

Purpose & Scope

The objective of this Compliance Product Guide is to accelerate customer understanding of effective DevSecOps processes and best practices against industry standards and frameworks. As part of this analysis, the JFrog Platform was evaluated against NIST SP 800-218, Secured Software Development Framework (SSDF) 1.1 from a product capability perspective. This document maps specific JFrog functionality to NIST 800-218 recommendations.

SSDF practices can be incorporated into each phase of the software development lifecycle, building a security practice that binds people, processes, and technology. This practice helps core software developers, mission application development, and the core DevSecOps community implement a rigid security framework — from finding vulnerabilities at early stages (aka shift left) to discovering new undetected vulnerabilities (zero days) and preventing future recurrences.

JFrog plays a critical role in securing each software supply chain, including the software development lifecycle, from code creation to artifacts in the runtime by four core tenets. JFrog's Advanced Security capabilities protect the application, its micro-services, and the cloud infrastructure configurations from causing exposures in production.

FOUR CORE TENETS

- DevOps
- DevSecOps
- DevGovOps
- AgentOps

RELATED FEDERAL GUIDANCE

NIST SP 800-218 also supplies requirements found in other standards, including CMMC, FedRAMP, DFARS, and FISMA, and underpins OMB memoranda M-22-16 and M-26-05.

SECTION 03

NIST SP 800-218 (SSDF) Overview

What is NIST SP 800-218?

NIST 800-218 describes a set of fundamental, sound practices for secure software development called the Secure Software Development Framework (SSDF). Organizations should integrate the SSDF throughout their existing software development practices, express their secure software development requirements to third-party suppliers using SSDF conventions, and acquire software that meets the practices described in the SSDF.

How can 800-218 be implemented?

The SSDF defines only a high-level subset of what organizations may need to do, so organizations should consult the references and other resources for additional information on implementing the practices. Not all practices apply to all use cases; organizations should adopt a risk-based approach to determine what practices are relevant, appropriate, and effective to mitigate the threats to their software development practices.

NIST 800-218 and EO 14028

The President's Executive Order (EO) on "Improving the Nation's Cybersecurity (14028)" issued on May 12, 2021 [EO14028], charged multiple agencies — including NIST — with enhancing cybersecurity through a variety of initiatives related to the security and integrity of the software supply chain.

Section 4 of the EO directed NIST to solicit input from the private sector, academia, government agencies, and others and to identify existing or develop new standards, tools, best practices, and other guidelines to enhance software supply chain security. Many subsections from Section 4e of the EO map to SSDF practices and tasks that can help address each subsection as part of a risk-based approach.

SECTION 04

The JFrog Platform Capabilities

Organizations providing DevSecOps capabilities to the US Public Sector need to practice and implement the security guidelines recommended in the NIST SP 800-218 SSDF. The JFrog Software Supply Chain Platform is purpose-built to help federal agencies operationalize SSDF requirements across the full software lifecycle. As the single source of record for every binary, container, and AI asset, the platform delivers deep binary and vulnerability analysis, cryptographic evidence collection, AI asset governance, and defense-grade delivery to air-gapped and classified networks. More detailed JFrog product mapping in Appendix A.

Universal Package Management

One trusted source of record for every binary, so teams know exactly what was built, from what, and where it runs.

Advanced Security for DevOps

Vulnerabilities, secrets, and misconfigurations caught from the IDE through runtime, prioritized by what is actually exploitable.

Application Risk Governance

Evidence-based policy gates that prove release integrity across the SDLC and shorten audit and authorization cycles.

Updates from Code to Device

Trusted updates delivered privately and securely all the way to the tactical edge and connected devices.

Truly Hybrid & Multi-Cloud

Deploy self-hosted, in the cloud, or fully air-gapped without sacrificing speed, availability, or control.

Scale DevSecOps to Infinity

Scale with mission demand across distributed sites and classifications while governance stays centralized.

APPLICABLE NIST 800-218 REQUIREMENTS

PO.1	PO.2	PO.3	PO.4	PO.5
PS.1	PS.2	PS.3	PS.4	PS.5
PW.1	PW.2	PW.4	PW.5	PW.6
PW.7	PW.8	PW.9	RV.1	RV.2

SECTION 05

Product Overview

The JFrog Software Supply Chain Platform gives agencies end-to-end visibility, security, and control to automate delivery of trusted releases, anchoring governance to the compiled binary as the system of record. Capabilities are grouped below by the operating model they serve.

DevOps

Manage every asset in the agency software supply chain — binaries, containers, AI/ML models, skills, and plugins — with attestation and build metadata that stand up to audit, across cloud, self-hosted, and air-gapped environments.

JFrog Artifactory	**Universal Artifact Management:** Curate a secure, single source of record for DevSecOps and the entire software supply chain, and update software across any environment with native support for 60+ package technology types and 100+ integrations.
JFrog Distribution	**Updates from Code to Devices:** Create your own private, fast, secure, hybrid distribution model that carries signed, immutable release bundles all the way to the tactical edge.
JFrog Connect	**Connected Device Management:** Extend the same governance to deployed and IoT devices in the field, so systems at the edge stay current, patched, and accounted for.

AgentOps

Secure, govern, and enforce every AI asset your coding agents consume as native artifacts in the supply chain, bringing shadow AI under the same controls that already govern mission software.

JFrog AI Catalog	**AI Asset Governance:** Govern the full AI supply chain, with models, MCP servers, and agent skills vetted, versioned, and stored as native artifacts alongside your binaries through the MCP Registry and Agent Skills Registry.

DevGovOps

Enforce policy as code and capture signed evidence on every release, so authorization packages, audits, and continuous monitoring draw on the same system of record as the binary itself.

JFrog AppTrust	**Evidence Collection and Application Risk Governance:** Capture and manage cryptographically signed attestation evidence needed for governed, production-ready releases and simplified audits, with evidence-based policies enforced at every quality gate.

DevSecOps

Secure the software supply chain at frontier AI speed. Prevent, detect, prioritize, and heal vulnerabilities across code, binaries, and AI/ML models automatically, so every release into a federal environment is trusted, verified, and compliant with SSDF practice.

JFrog Xray	**Software Composition Analysis:** Identify and resolve security vulnerabilities and license compliance issues in your open-source dependencies across every build.
JFrog Xray	**Malicious Package Detection:** Discover and eliminate unwanted or unexpected packages using a unique database of thousands of malicious packages identified by the JFrog security research team and continuously aggregated from global sources.
JFrog Curation	**Perimeter Defense:** Block malicious or unwanted OSS packages and AI assets before they reach developers, vetting every package against your security, compliance, and operational policies at the gate.
JFrog IDE plugins, CLI, and Frogbot	**Shift as Far Left as Possible:** Surface vulnerabilities, license violations, remediation options, and applicability context inside the IDE, with dependency, container, and on-demand scans automated in build workflows.
JFrog Xray	**SBOM and Regulatory Compliance:** Generate SPDX, CycloneDX, and VEX standard-format SBOMs with binary-level accuracy that goes well beyond metadata requirements, and automate publication of SBOMs and associated CVEs.
JFrog Xray and Curation	**FOSS Compliance and License Clearing:** Monitor, control, and validate the licenses you ship, with automated license clearing that keeps development teams on fully approved components.
JFrog Advanced Security	**Contextual Analysis and Remediation:** Find and fix exposed secrets, OSS vulnerabilities, IaC and container security issues across the supply chain, with contextual analysis that eliminates false positives and concrete remediation guidance.
JFrog Runtime	**Runtime Visibility:** Extend governance to the binary in production across self-hosted, cloud, multi-cloud, and air-gapped deployment targets, without sacrificing speed or availability.

SECTION 06

Why This Matters for Federal Agencies

Federal agencies and other enterprises continue to face security threats such as cyberattacks, ransomware, security vulnerabilities in legacy software, and more. NIST SP 800-218 guides on building strong secured software using the practices and guidelines specified in the SSDF framework.

Adopting the JFrog Software Supply Chain Platform — including **JFrog AppTrust** for Policy-as-Code governance and **JFrog AI Catalog** for AI asset governance — helps federal agencies and other enterprises achieve compliance with NIST SP 800-218 and other security regulations such as OMB M-22-16 and OMB M-26-05. JFrog Security helps these organizations with advanced security protection in place to defend against ever-evolving cyber threats.

19

SSDF practices mapped to JFrog Platform capability in Appendix A

FULL

Inheritance level assessed across every mapped requirement

60+

Package technology types natively supported, plus 100+ integrations

APPENDIX A

Security Requirements Mapping

NIST SP 800-218 (Full/Partial) — JFrog Platform capability by SSDF practice.

REQ. ID	SECURITY REQUIREMENT	INHERIT.	JFROG PLATFORM CAPABILITY
PO.1	Define Security Requirements for Software Development	FULL	The JFrog Platform manages all component vulnerability policies, FOSS licensing policies, and FOSS operational policies. Policies can be applied to every SDLC process across the organization, from providing early insights to developers (in the IDE and CLI) up to blocking releases, all in a fully automated fashion.
PO.2	Implement Roles and Responsibilities	FULL	The JFrog Platform has a flexible Roles mechanism to support the central administration of security policies with full delegation to teams and projects where needed. It also features Role Based Access Controls (RBAC), ensuring access to certain capabilities only resides with the correct teams and roles.
PO.3	Implement Supporting Toolchains	FULL	The JFrog Platform allows for the automation of security functions across every stage of the SDLC. These include other Security and DevOps tools, ensuring automated implementation of security practices across tool chains and processes. Open APIs, built-in integrations, support for custom plugins, and the JFrog CLI ensure automation securely across tool chains for all SDLC phases.
PO.4	Define and Use Criteria for Software Security Checks	FULL	The JFrog Platform provides the ability to define security policies to evaluate artifacts, builds, or container images across all stages of the SDLC. These policies can take the form of security vulnerabilities, operational risk, license compliance, and malicious packages. JFrog Curation blocks non-compliant open-source components before they enter the development environment, and JFrog AI Catalog applies the same criteria to models, MCP servers, and agent skills. During the development lifecycle, evidence is gathered and stored, providing auditability and security at each stage of the SDLC.

REQ. ID	SECURITY REQUIREMENT	INHERIT.	JFROG PLATFORM CAPABILITY
PO.5	Implement and Maintain Secure Environments for Software Development	FULL	• The Platform can be deployed in its own environmental boundary (network, access) with uniform access control and authorization. It is implemented as microservices, guaranteeing a level of trust and defined access and authorization boundaries between services. • "Projects" isolate resources such as artifacts, builds, and repositories with RBAC. A request router directs each request to the right service — Artifactory, Access, Replicator, Event, Observability, Integration, Metadata, and JFrog Connect — with additional routers for Xray, AppTrust, and Distribution. • Support, build, test, dev, and production environments are protected by user authentication — multi-factor, SSO, SAML — backed by your identity manager, plus fine-grained permissions and RBAC over artifacts and builds. • Each instance has well-defined network segmentation (dedicated VPC, network adapter) with defined ingress and egress ports and security groups, and can be deployed air-gapped with zero ingress and egress. TLS protects every ingress and all service-to-service communication, with Access acting as the CA signing certificates for all Platform services. • Continuous monitoring provides uniform logs and metrics per deployment plus audit-trail records of all user, group, and permission operations. JFrog AppTrust captures signed evidence at every stage, Xray and Advanced Security continuously monitor artifacts, and Distribution's immutable Release Bundles are hash-verified on receipt.
PS.1	Protect All Forms of Code from Unauthorized Access and Tampering	FULL	The JFrog Platform allows organizations to create immutable release assets immediately from the output of the build process. It stores artifact metadata and configuration files and creates buildInfo comprising the metadata required to recreate any build. Any changes to artifacts or repositories are noted for auditability, and RBAC ensures access to certain capabilities only resides with the correct teams and roles. Artifactory stores artifacts using checksum-based storage, avoiding duplicate files and optimizing binary management and protection. Distribution creates immutable Release Bundles, signed and encrypted with GPG keys, which the target Platform hash-verifies on receipt.

REQ. ID	SECURITY REQUIREMENT	INHERIT.	JFROG PLATFORM CAPABILITY
PS.2	Provide a Mechanism for Verifying Software Release Integrity	FULL	The JFrog Platform ensures the integrity of software releases by defining a potential release as an immutable asset early in the SDLC, capturing evidence of all actions taken against it as it matures, and leveraging policy and quality gates to automate or block advancement toward consumption. JFrog AppTrust captures cryptographically signed evidence at each stage, attesting to the authenticity and provenance of your binaries as they progress through the SDLC. Artifactory enables signing for package types such as RPM, Maven, and Debian repositories; keys are generated using GPG so a package Release file cannot be manipulated from the generated metadata. Distribution's immutable Release Bundles are signed and encrypted with GPG keys and hash-verified on receipt.
PS.3	Archive and Protect Each Software Release	FULL	Artifactory protects artifacts stored in repositories with role-based access control and a fine-grained permission model that prevents unauthorized access to binaries, builds, configuration, and metadata files. Distribution creates Release Bundles signed and encrypted with GPG keys. Xray generates a Software Bill of Materials (SBOM) — enabling DevSecOps engineers to understand and analyze the open-source components and dependencies of their builds, providing provenance data — exportable in SPDX and CycloneDX formats and updated on every build. Xray can be configured with security policies for fail-build/blocking, continuous scanning, and traceability with ascendants and descendants. JFrog AppTrust evidence attests to the authenticity of your binaries, and Artifactory cold storage optimizes storage of archived artifacts and builds for regulatory purposes, with the ability to restore them preserving paths and repositories.

REQ. ID	SECURITY REQUIREMENT	INHERIT.	JFROG PLATFORM CAPABILITY
PW.1	Design Software to Meet Security Requirements and Mitigate Security Risks	FULL	JFrog provides a single platform to define and apply application security and compliance policies across the SDLC, with the ability to allow exceptions where required. It provides detailed vulnerability reports and insights to enable security risk assessments and identifies the impact of vulnerabilities with traceability. JFrog Curation blocks malicious or risky packages from entering your software supply chain, and JFrog AI Catalog governs models, MCP servers, and agent skills to the same standard — radically reducing risk in development. Detailed remediation guidance reaches developers in their IDE, CLI, or platform UI, covering OSS vulnerabilities, exposed secrets, misconfigurations, and poor coding that could lead to exposures. Distribution creates GPG-signed Release Bundles, Xray generates an SBOM providing provenance, and JFrog AppTrust attests to the authenticity of your binaries and builds with signed evidence.
PW.2	Review the Software Design to Verify Compliance with Security Requirements and Risk Information	FULL	Artifactory protects artifacts stored in repositories with role-based access control and a fine-grained permission model that prevents unauthorized access to binaries, configuration, and metadata files. Distribution creates GPG-signed Release Bundles. Xray generates an SBOM — enabling DevSecOps engineers to understand and analyze the open-source components and dependencies of their builds, providing provenance data — exportable in SPDX and CycloneDX (VEX) format and updated on every build. Xray can be configured with security policies for fail-build/blocking, continuous scanning, and traceability with ascendants and descendants, and JFrog AppTrust attests to the authenticity of your binaries and builds with signed evidence.

REQ. ID	SECURITY REQUIREMENT	INHERIT.	JFROG PLATFORM CAPABILITY
PW.4	Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality	FULL	• JFrog Curation policies control and prevent malicious and risky third-party packages from entering your supply chain, filtering package downloads against a defined set of policy templates, with JFrog AI Catalog extending the same vetting to models, MCP servers, and agent skills. • Automate Curation workflows for a transparent, seamless developer experience. Curate OSS libraries in an Artifactory instance in a DMZ network segment, isolating packages in sanctioned repositories, then promote them back to the developer instance using Federated repositories or Distribution for air-gapped environments. • JFrog Curation is the source of truth for approved packages — OSS and proprietary — with private metadata for use within the organization. Promotion of artifacts can be achieved with build promotion between environments or with immutable Release Bundles between instances, including unclassified to classified. • JFrog Xray generates an SBOM providing provenance data, and adds malicious package detection, vulnerability detection, OSS license checks, and operational risk scoring for detected components — factoring in version age, EOL project detection, and maintenance level from public open-source data. • JFrog Advanced Security manages application exposures including SAST, IaC security, exposed secrets, service and OSS library misconfigurations, and CVE applicability in context. Artifactory cold storage retains artifacts that are no longer in use but cannot be deleted under archiving regulations.
PW.5	Create Source Code by Adhering to Secure Coding Practices	FULL	JFrog SAST enables development teams to write and commit trusted source code. Security-focused engines deliver scans that detect zero-day security vulnerabilities and check code against unsafe data flow, functions and calls, and other known bad coding practices. JFrog's IDE plugin warns developers of source code vulnerabilities at development time, and can also inspect source code in your Git repository — a strong shift-left practice.

REQ. ID	SECURITY REQUIREMENT	INHERIT.	JFROG PLATFORM CAPABILITY
PW.6	Configure the Compilation, Interpreter, and Build Processes to Improve Executable Security	FULL	JFrog's IDE plugin warns developers of source code, open-source, and configuration vulnerabilities at development time as the software is being built. Knowing the open source and source code are trusted at compilation time helps ensure executable (binary) security. JFrog also runs security, license, configuration, and secrets scans on binaries throughout the SDLC, ensuring binaries are secure before deployment and in production.
PW.7	Review and/or Analyze Human-Readable Code to Identify Vulnerabilities and Verify Compliance	FULL	JFrog SAST enables development teams to write and commit trusted source code. Security-focused engines deliver scans that detect zero-day security vulnerabilities and check code against unsafe data flow, functions and calls, and other known bad coding practices. It checks code in the IDE and in Git repositories.
PW.8	Test Executable Code to Identify Vulnerabilities and Verify Compliance	FULL	JFrog's security scans binaries for proprietary, open-source, configuration, and service vulnerabilities throughout the software supply chain. Automate security and compliance policies to detect and remediate from your Git repository to production. This approach validates the security of the binaries that will execute in runtime, and continuous scanning detects and remediates vulnerabilities in runtime.
PW.9	Configure Software to Have Secure Settings by Default	FULL	JFrog's Advanced Security focuses on binary scanning to detect and remediate security, configuration, micro-service, secrets, and Infrastructure as Code (IaC) vulnerabilities. JFrog checks IaC, OSS libraries, and service configurations to make sure they follow best practices and don't pose a security risk. Scans are performed in source code, configuration files, and inside binaries across the stages of the SDLC.

REQ. ID	SECURITY REQUIREMENT	INHERIT.	JFROG PLATFORM CAPABILITY
RV.1	Identify and Confirm Vulnerabilities on an Ongoing Basis	FULL	JFrog scans from code to runtime. JFrog's security research team continuously updates the vulnerability database through its own research and the inclusion of public (NVD) and private third-party sources. Their attack vector and CVE research yields easy-to-use remediation for speedy fixes by developers or security teams. Artifacts including binaries are continuously scanned for known vulnerabilities with automated policies on an ongoing basis, and JFrog Curation and JFrog AI Catalog re-evaluate consumed packages and AI assets as new findings emerge.
RV.2	Assess, Prioritize, and Remediate Vulnerabilities	FULL	JFrog's Advanced Security contextual analysis assesses the applicability of identified CVEs by analyzing their context and attributes — checking whether first-party code calls the vulnerable function, whether CVEs sit in reachability paths, and what configurations and file attributes apply — enabling prioritization. JFrog's database coupled with contextual analysis recommends concrete, actionable, cost-effective remediation steps that account for the specific attributes and configurations of the application, often a simple code change. Security insights dashboards enable easy high-level prioritization at build, project, or application level, and applicability analysis reaches development teams through shift-left integration in all popular IDEs, the JFrog CLI, and JFrog Frogbot with Git repository integration and scanning. IaC, misconfiguration, and secrets findings are analyzed alongside, with detailed remediation advice from the security research team for high-severity findings.
RV.3	Analyze Vulnerabilities to Identify Their Root Causes	FULL	JFrog also checks Infrastructure as Code (IaC), library and service configurations, and exposed secrets. For high-severity findings, JFrog provides detailed remediation advice from its security research team.



For a detailed platform walkthrough or an agency-specific SSDF assessment, contact your JFrog Public Sector team.

Get Started with JFrog