DEVOPS SOLUTIONS · SECURE SOFTWARE PIPELINE · NIST SP 800-218 V1.1

SSDF coverage map

The NIST Secure Software Development Framework lists **42 tasks** across four practice groups. This map shows how each one is handled when we deploy our secure pipeline: what the pipeline does on its own, where it enforces a policy you set, where we hand you a ready template, and what stays with your team.

All 42 SSDF tasks

Select a category to highlight it. Select a cell to see the task.

Built into the pipeline 13 | Pipeline enforces your policy 10 | Template provided 9 | Your team 10

PO · 13 tasks
Prepare the organization

PO.1.1 PO.1.2 PO.1.3 PO.2.1 PO.2.2 PO.2.3 PO.3.1 PO.3.2 PO.3.3 PO.4.1
PO.4.2 PO.5.1 PO.5.2

PS · 4 tasks
Protect the software

PS.1.1 PS.2.1 PS.3.1 PS.3.2

PW · 16 tasks
Produce well-secured software

PW.1.1 PW.1.2 PW.1.3 PW.2.1 PW.4.1 PW.4.2 PW.4.4 PW.5.1 PW.6.1 PW.6.2
PW.7.1 PW.7.2 PW.8.1 PW.8.2 PW.9.1 PW.9.2

RV · 9 tasks
Respond to vulnerabilities

RV.1.1 RV.1.2 RV.1.3 RV.2.1 RV.2.2 RV.3.1 RV.3.2 RV.3.3 RV.3.4

13
Built into the pipeline

10
Pipeline enforces your policy

9
Template provided

10
Your team

Evidence collected by the pipeline

SSDF asks for artifacts that show each practice is actually followed. The pipeline produces them as a side effect of normal work, so preparing for a customer or auditor review means exporting a folder, not writing a report.

SSDF describes outcomes and leaves the choice of tools to you. NIST does not certify against SSDF; the framework is used to describe practices to customers and suppliers.

- **Review history for every change**
  Who approved what and when, with protected branches and code owners.
- **Scanner results per release**
  Static analysis, dependency analysis, secrets detection, dynamic tests, with triage decisions.
- **SBOM for every release**
  In a standard format, stored with the release and monitored for new vulnerabilities.
- **Signed artifacts and checksums**
  So your customers can verify a release has not been tampered with.
- **Build logs and tool versions**
  Kept read-only, together with the pipeline configuration as code.
- **Register of accepted exceptions**
  Every waived finding, with the reason and the person who accepted it.

PO Prepare the organization

13 tasks

TASK	WHAT IT COVERS	HANDLED BY
PO.1.1	Document security requirements for development infrastructure and processes	Template provided
PO.1.2	Document security requirements for the software you build	Pipeline + your policy
PO.1.3	Pass security requirements on to suppliers of commercial components	Your team
PO.2.1	Define roles and responsibilities across the development life cycle	Template provided
PO.2.2	Provide role-based training	Your team
PO.2.3	Secure management commitment to secure development	Your team
PO.3.1	Specify which tools each toolchain must include and how they integrate	Built in
PO.3.2	Deploy, operate and maintain tools securely, as code	Built in
PO.3.3	Configure tools to generate evidence of secure practices	Built in
PO.4.1	Define criteria for software security checks	Pipeline + your policy
PO.4.2	Gather and protect the information behind those criteria	Built in
PO.5.1	Separate and protect development, build, test and distribution environments	Pipeline + your policy
PO.5.2	Harden developer workstations	Your team

PS Protect the software

4 tasks

TASK	WHAT IT COVERS	HANDLED BY
PS.1.1	Store all code with least-privilege access and full change history	Built in
PS.2.1	Give customers a way to verify release integrity	Built in
PS.3.1	Archive and protect each release	Built in
PS.3.2	Keep provenance data (SBOM) for every component of each release	Built in

PW Produce well-secured software

16 tasks

TASK	WHAT IT COVERS	HANDLED BY
PW.1.1	Model threats and assess security risk	Template provided
PW.1.2	Track security requirements, risks and design decisions	Pipeline + your policy
PW.1.3	Use standard security features instead of custom ones	Your team
PW.2.1	Review the design against security requirements	Template provided
PW.4.1	Acquire and maintain well-secured third-party components	Built in
PW.4.2	Build well-secured shared components in-house	Your team
PW.4.4	Verify third-party components throughout their life cycle	Built in
PW.5.1	Follow secure coding practices	Pipeline + your policy
PW.6.1	Use current, verified compilers and build tools	Built in
PW.6.2	Use secure compiler and build settings	Pipeline + your policy
PW.7.1	Decide where code review and code analysis apply	Template provided
PW.7.2	Review and analyse code, and triage every finding	Built in
PW.8.1	Decide which executable testing is needed	Template provided
PW.8.2	Run security tests and record the results	Pipeline + your policy
PW.9.1	Define secure default settings	Your team
PW.9.2	Implement and document the default settings	Your team

RV Respond to vulnerabilities

9 tasks

TASK	WHAT IT COVERS	HANDLED BY
RV.1.1	Collect vulnerability reports and monitor public sources	Pipeline + your policy
RV.1.2	Re-analyse supported releases on a regular schedule	Built in
RV.1.3	Set a vulnerability disclosure and response policy	Template provided
RV.2.1	Analyse each vulnerability and assess its risk	Pipeline + your policy
RV.2.2	Plan and deliver the response, including advisories	Your team
RV.3.1	Find the root cause of each vulnerability	Template provided
RV.3.2	Analyse root causes over time to spot patterns	Template provided
RV.3.3	Search for similar vulnerabilities proactively	Pipeline + your policy
RV.3.4	Update the development process so root causes do not recur	Your team

Source: NIST Special Publication 800-218, Secure Software Development Framework (SSDF) Version 1.1, February 2022, doi.org/10.6028/NIST.SP.800-218. Task names are shortened; the publication has the full wording. A revision (version 1.2) was released as a public draft in December 2025; this map will be updated when the final version is published.

The split of responsibilities is our standard package. The actual scope for your organisation is fixed after a survey of your environment.