

# Testplan

## First Collection of Tasks

This document gives an overview of the required tasks for the security testing of the GAIA-X Authentication & Authorization Service.

## Security Testing Requirments

Regarding the required security testing the documentation “Specification of non-functional Requirements for Gaia-X Federation Services Security and Privacy by Design NF.SPBD” (GXFS\_Nonfunctional\_Requirements\_SPBD.pdf) specifies the following:

### [Section 3.4.4 - Testing and Acceptance \(Page 26\)](#)

#### *Planning security tests*

Policies and procedures are required to ensure the timely identification and addressing of vulnerabilities.

The following aspects must be covered:

- frequency of tests (regular, continuous, event driven)
- scope of tests
  - Code Testing
  - Vulnerability Scanning
  - Penetration Testing
  - Compliance Testing
- test types, test environments and test tools (e.g. on- vs. offsite tests, tests in test vs. production environments, integration of security tools into build processes)
- documentation of test results (e.g. automated generation and delivery of test reports as required documents for the Compliance Documentation Service [6])
- remediation of test findings

Further on page 27 and 28 the required tests are described in detail:

### [Subsection Performing security tests \(Section 3.4.4, page 27/28\)](#)

#### *Development/build phase*

- Human-based code review
  - Peer review of code through different developers and code signoff shall be performed.
- Tool based source code review.
  - Automatic tests must be performed to detect secrets such as API keys, private crypto keys etc. in the source code or packaged applications (e.g. container).
  - Software composition analysis (SCA) must be performed to ensure that libraries, dependencies, and other 3rd party artifacts, is used to properly identify, document and check (security & software license compliance) the code base.
  - Static source code testing (SAST) must be integrated into the development cycle (ideally in the CI/CD pipeline) and code is mark non-compliant if it fails automatic checks (build fail in case of non-compliance).

- Dynamic source code testing (DAST) tools shall be included into the development process (or CI/CD pipeline) or may be executed as part of the penetration test.
    - Should most likely be run as part of the PenTest (Suggestion by Wolfgang)
- Tool based vulnerability analysis
  - Any software dependency that are necessary to run the code must be checked for vulnerabilities. This includes executables, library, container base images and overlay files as well as other dependencies or artifacts.
- Configuration Compliance
  - Configuration of software used must be hardened based on best practice requirements (such as CIS). The correct implementation of these hardening requirements needs to be (automatically) checked.
- Security (Unit) Tests
  - Security function/controls used in the code (e.g. authentication) must be verified with test cases. Tests should be integrated into automatic testing wherever possible. New test cases are developed if new security controls are integrated.
- Reporting
  - The execution and results of all security tests must be documented and/or logged. In case of follow up activities (e.g. in case of a failed security check), these activities as well as the outcome is also documented. It is ensured, that all security deviations are properly and timely addressed, this also includes documentation of false positives or “won’t fix” items (incl. their justification). Reports, test concepts and individual test descriptions are available to Gaia-X on request (incl. auditing purposes to ensure good security practices).

#### *Test phase*

- Penetration tests must be performed as followed (see also [7] OPS-19.2 - 6) In contrast to vulnerability scans, penetration tests are deeper and more specific. Beside generic automated tests (used by vulnerability scanners) the penetration tester must also take into consideration Federation Service specific test use cases. Where vulnerability scans for e.g. missing patches, a penetration test tries to exploit known vulnerabilities to start a deeper investigation with the aim to identify further weaknesses of the service.
  - Testing:
    - A penetration test must be performed before the initial go live of the Federation Service. This test is mandatory to get the service approval.
    - In case of major service updates further penetration tests shall be performed.
    - The penetration tests should focus on aspects not covered by (automatic) testing during development, such as business logic flaws.
  - Report
    - The penetration test report must be delivered to the responsible entity.

## Security Testing Tasks & Tooling

Based on the requirements cited in the previous chapter following security testing tasks can be derived:

### Development/build phase

ID	SECT-1
Name	Static Code Analysis
Scope of Test	Code Test
Frequency of Tests	Event Driven – Triggered by pushing code to the repository
Integration	integrated into the CI/CD pipeline
Test environment	development environment
Tooling	SonarQube
Objective	Static source code testing (SAST) must be integrated into the development cycle (ideally in the CI/CD pipeline) and code is mark non-compliant if it fails automatic checks (build fail in case of non-compliance).
Remediation of Test Findings	build process stops after findings, developer needs to fix issued before the code can be released or merged into the master
Storage of Reports	tbd
Process	tbd

ID	SECT-2
Name	Software composition analysis - security
Scope of Test	Vulnerability Scanning
Frequency of Tests	Event Driven – Triggered by pushing code to the repository
Integration	integrated into the CI/CD pipeline
Test environment	development environment
Tooling	Dependency Check
Objective	Software composition analysis (SCA) must be performed to ensure that libraries, dependencies, and other 3rd party artifacts, do not have any currently known vulnerabilities.
Remediation of Test Findings	build process stops after findings, developer needs to fix issued before the code can be released or merged into the master
Storage of Reports	Reports will be stored in the Gitlab repository
Process	tbd

ID	SECT-3
Name	Software composition analysis - software license compliance
Scope of Test	License Scanning
Frequency of Tests	Regular – this test is run manually
Integration	No integration
Test environment	local development machines
Tooling	LicencePlugin
Objective	Software composition analysis (SCA) must be performed to ensure that libraries, dependencies, and other 3rd party artifacts comply with the OSS license in use.
Remediation of Test Findings	If the license of a library, dependency, and other 3rd party artifact is incompatible with the used OSS license the specific software can not be used and needs to be replaced with an compliant solution.

Storage of Reports	Reports will be stored in the Gitlab repository
Process	tbd

ID	SECT-4
Name	Detect Secrets
Scope of Test	Code Test
Frequency of Tests	Event Driven – Triggered by pushing code to the repository
Integration	integrated into the CI/CD pipeline
Test environment	development environment
Tooling	TruffleHog
Objective	Automatic tests must be performed to detect secrets such as API keys, private crypto keys etc. in the source code or packaged applications (e.g. container).
Remediation of Test Findings	build process stops after findings, developer needs to fix issued before the code can be released or merged into the master
Storage of Reports	tbd
Process	tbd

ID	SECT-5
Name	Configuration Compliance
Scope of Test	Compliance Testing
Frequency of Tests	Event Driven – Triggered by pushing code to the repository
Integration	integrated into the CI/CD pipeline
Test environment	development environment
Tooling	Checkov
Objective	Configuration of software used must be hardened based on best practice requirements (such as CIS). The correct implementation of these hardening requirements needs to be (automatically) checked.
Remediation of Test Findings	build process stops after findings, developer needs to fix issued before the code can be released or merged into the master
Storage of Reports	tbd
Process	tbd

ID	SECT-6
Name	Security (Unit) Tests
Scope of Test	Code Test
Frequency of Tests	Event Driven – Triggered by pushing code to the repository or by building locally
Integration	integrated into the CI/CD pipeline and local build process
Test environment	development environment and local development machines
Tooling	Unit Testing Framework
Objective	Security function/controls used in the code (e.g. authentication) must be verified with test cases. Tests should be integrated into automatic testing wherever possible. New test cases are developed if new security controls are integrated.
Remediation of Test Findings	build process stops after findings, developer needs to fix issued before the code can be released or merged into the master
Storage of Reports	Reports are automatically documented in the Gitlab for each push

Process	tbd
---------	-----

ID	SECT-7
Name	Code Review – Before Merge
Scope of Test	Code Test
Frequency of Tests	Event Driven – Triggered by merge request
Integration	No integration
Test environment	-
Tooling	Gitlab
Objective	Peer review of code through different developers and code signoff shall be performed.
Remediation of Test Findings	A branch can be merged into the master only after a successful code review.
Storage of Reports	Reports will be stored in the Gitlab repository
Process	tbd

ID	SECT-8
Name	Code Review – Security Review
Scope of Test	Code Test
Frequency of Tests	Regular – after change to used OIDC implementation
Integration	No integration
Test environment	-
Tooling	-
Objective	The used OIDC flows of the employed OIDC implementation must be checked for compliance with the security measures proposed in the “OAuth 2.0 Threat Model and Security Considerations” <sup>1</sup> and the security considerations of the “Self-Issued OpenID Provider v2” <sup>2</sup> standard
Remediation of Test Findings	tbd
Storage of Reports	Report will be stored in the Gitlab repository
Process	tbd

## Test phase

ID	SECT-9
Name	Penetration Test
Scope of Test	Penetration Test
Frequency of Tests	A penetration test must be performed before the initial go live of the Federation Service.
Integration	No integration
Test environment	Pre-production environment
Tooling	-
Objective	In contrast to vulnerability scans, penetration tests are deeper and more specific. Beside generic automated tests (used by vulnerability scanners) the penetration tester must also take into consideration Federation Service specific test use cases. Where vulnerability scans for e.g. missing

<sup>1</sup> <https://datatracker.ietf.org/doc/html/rfc6819>

<sup>2</sup> [https://openid.net/specs/openid-connect-self-issued-v2-1\\_0.html](https://openid.net/specs/openid-connect-self-issued-v2-1_0.html)

	patches, a penetration test tries to exploit known vulnerabilities to start a deeper investigation with the aim to identify further weaknesses of the service.
Remediation of Test Findings	tbd
Storage of Reports	Report will be stored in the Gitlab repository must be delivered to the responsible GAIA-X entity
Process	tbd

ID	SECT-10
Name	Penetration Test - Dynamic source code testing
Scope of Test	Code Test
Frequency of Tests	A penetration test must be performed before the initial go live of the Federation Service.
Integration	No integration
Test environment	Pre-production environment
Tooling	-
Objective	Dynamic source code testing (DAST) tools shall be included into the development process (or CI/CD pipeline) or may be executed as part of the penetration test.
Remediation of Test Findings	tbd
Storage of Reports	Report will be stored in the Gitlab repository must be delivered to the responsible GAIA-X entity
Process	tbd