Fraunhofer
AISEC

Fraunhofer-Institut für Angewandte
und Integrierte Sicherheit AISEC

Christian Banse

Integrating the CSAF Standard into Dependency-Track with Kotlin-CSAF: Early Insights and Developments

About myself

@oxisto

Work Life

Head of Department “Service & Application Security” @ Fraunhofer AISEC

We developed multiple open-source projects in my department @ Fraunhofer AISEC, such as:

cpg: A library to generate code property graphs
Codyze: A static code analyzer for Java, C/C++, Go, Python
Clouditor: A tool for continuous cloud certification
kotlin-csaf: Kotlin implementation of CSAF standard + basic validator
csaf-rust: PoC for CSAF library in Rust
…

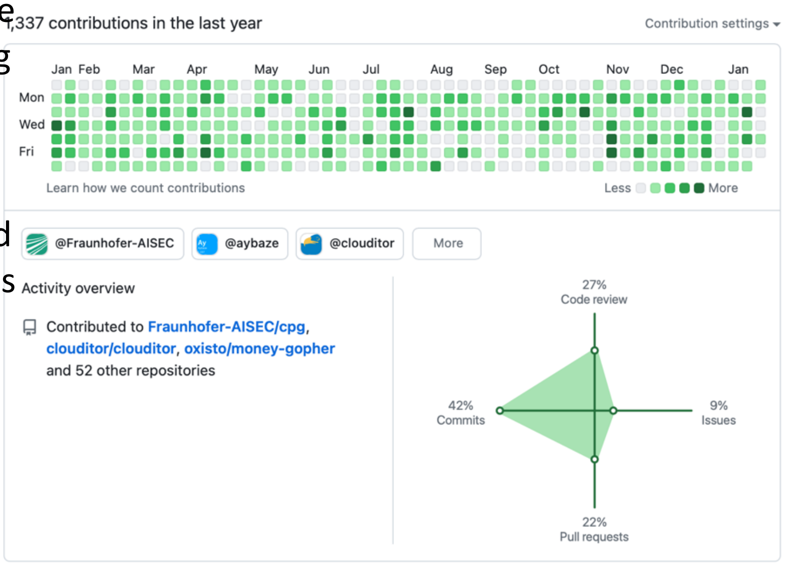
Fraunhofer AISEC also recently joined OASIS-OPEN as TC member for CSAF

Private Life

Avid supporter of Open-Source development (mainly on GitHub, @oxisto).
Main technologies: Go, TypeScript

Co-Maintainer of **jwt-go**, the largest Go library for parsing and validation of JSON Web Tokens (JWT).

A lot of never-to-be-finished projects from different areas (security, finance, …)



1,337 contributions in the last year

Contribution settings

Learn how we count contributions

Activity overview

Contributed to Fraunhofer-AISEC/cpg, clouditor/clouditor, oxisto/money-gopher and 52 other repositories

TLP:CLEAR

Project: Integrating CSAF into DependencyTrack

Contracted by BSI

Goals:

Integration of CSAF as Sources for Vulnerabilities in DependencyTrack
Implementation of a CSAF SBOM matching system (Section 9.1.17)

Side Goals

Implementation of a CSAF parsing library for the JVM

Development in the Open

Current state / forks of DT always available at https://github.com/csaf-sbom
First upstream PRs are planned (January 2025)

Fraunhofer AISEC

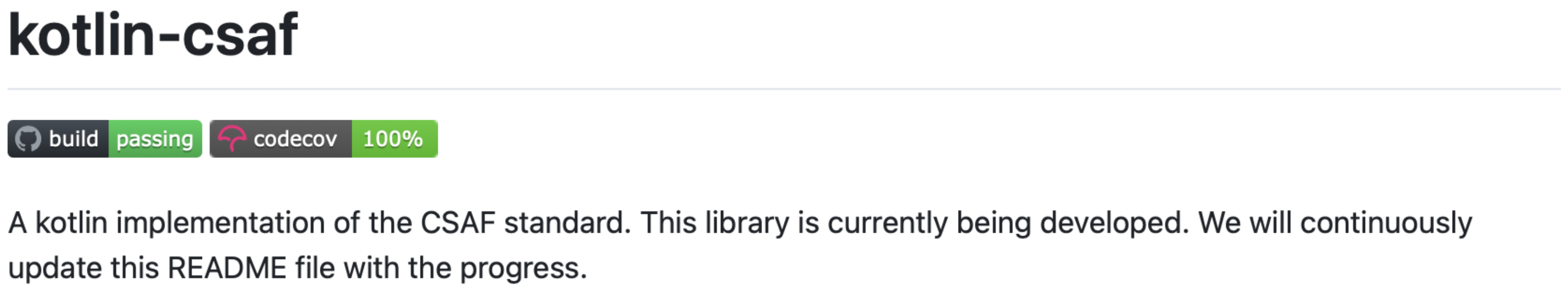
kotlin-csaf
Small Detour

There was no CSAF parsing / validation library available for Java (or the JVM)

→ We decided to implement "kotlin-csaf" (https://github.com/csaf-sbom/kotlin-csaf)

Why Kotlin

- It's just cooler than Java ;)
- Less boiler-plate, more efficient, good support for async programming patterns
- 100% compatible with JVM, but also multiplatform possible! → JVM, native, wasm, …
- Currently, the project is prepared for multiplatform, but the only target is JVM (for now)


kotlin-csaf

A kotlin implementation of the CSAF standard. This library is currently being developed. We will continuously update this README file with the progress.

kotlin-csaf

Features

Async Fetching API

- Fetch providers from aggregator
- Fetch provider from URL/domain
- Fetch (all) documents from provider
- Validate document (see below)

Blocking Fetching API for Java

- The same features are available

Validation API

- Currently, targeting conformance target “basic CSAF validator” → All mandatory tests are implemented
- **Also available as a CLI application to validate CSAF JSON files on disk**

kotlin-csaf

Next Steps

Matching API

Working to conformance target CSAF SBOM matching system

Input

- CSAF
- Generic SBOM format: ProtoPOM (supports CycloneDX, SPDX)

kotlin-csaf

Automation all the way

Data Classes

- Auto-Generation of Kotlin classes for CSAF aggregator metadata, CSAF provider metadata and CSAF document from JSON schema
- Classes are “validation-on-creation” → Constructors include verification code according to JSON schema

→ “Impossible” to create invalid CSAF data classes

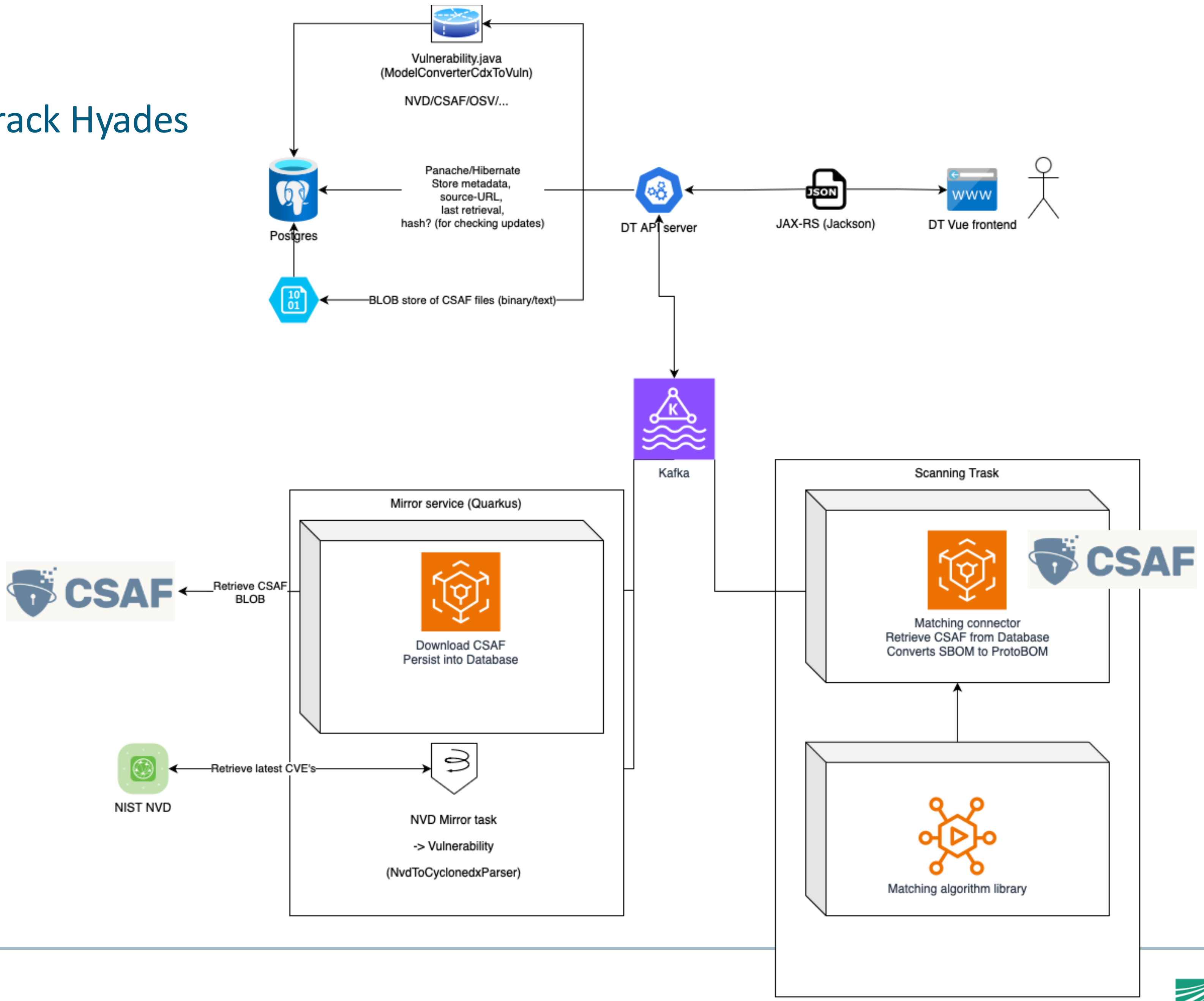
Testing

- Leveraging the CSAF test files in the TC repo via git submodule
- Automated update of TC repo via Dependabot
- Unit tests contain a check, whether all test files in the TC repo were “consumed” during testing
- → If new test files appear, our tests will fail, informing us that we need to include them

Fraunhofer AISEC

Architecture

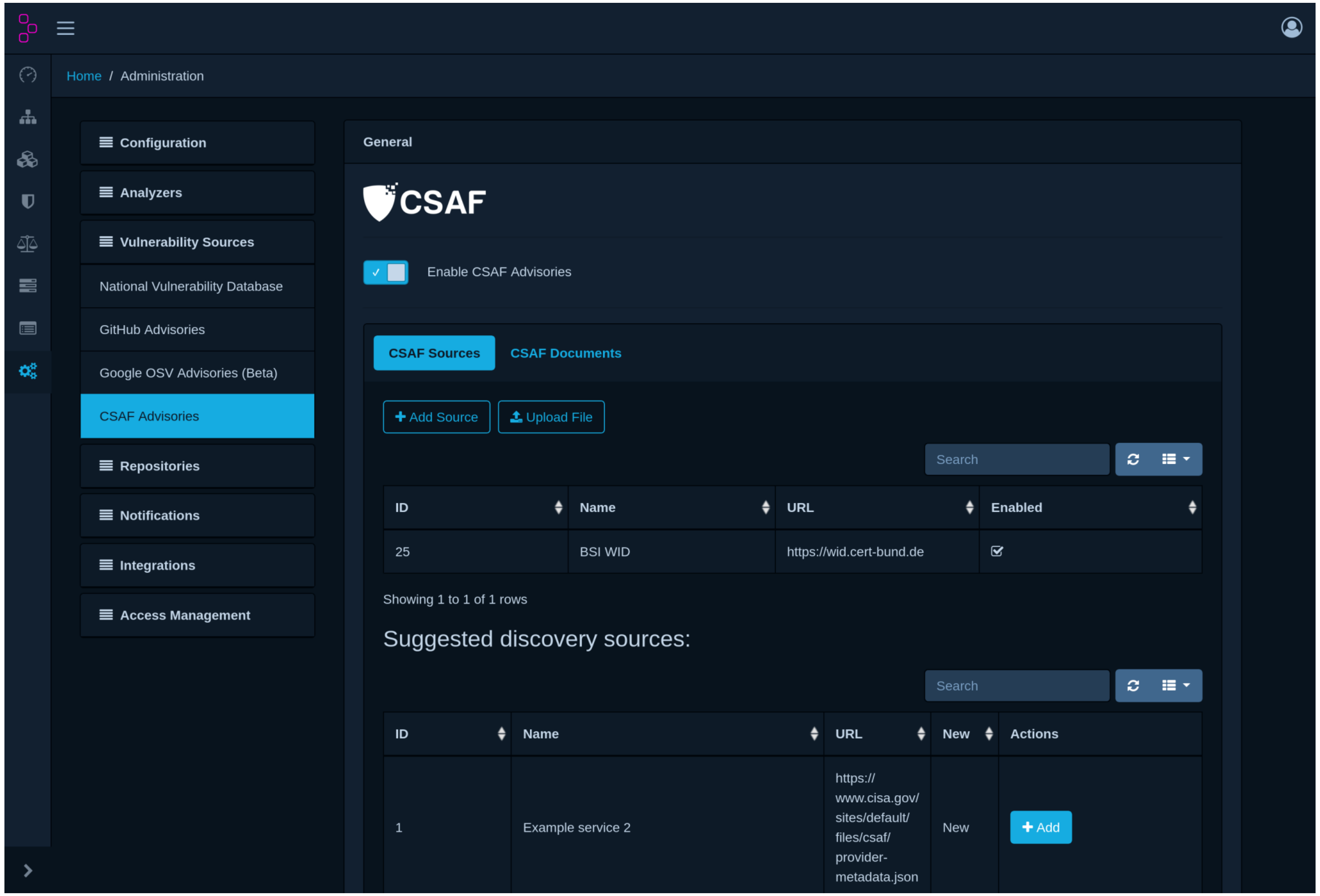
Based on DependencyTrack Hyades



UI Demo Screenshots

Fraunhofer AISEC

List of CSAF Sources



Home / Administration

- Configuration
- Analyzers
- Vulnerability Sources
  - National Vulnerability Database
  - GitHub Advisories
  - Google OSV Advisories (Beta)
  - CSAF Advisories
- Repositories
- Notifications
- Integrations
- Access Management

General

CSAF

✓ Enable CSAF Advisories

CSAF Sources | CSAF Documents

+ Add Source | Upload File

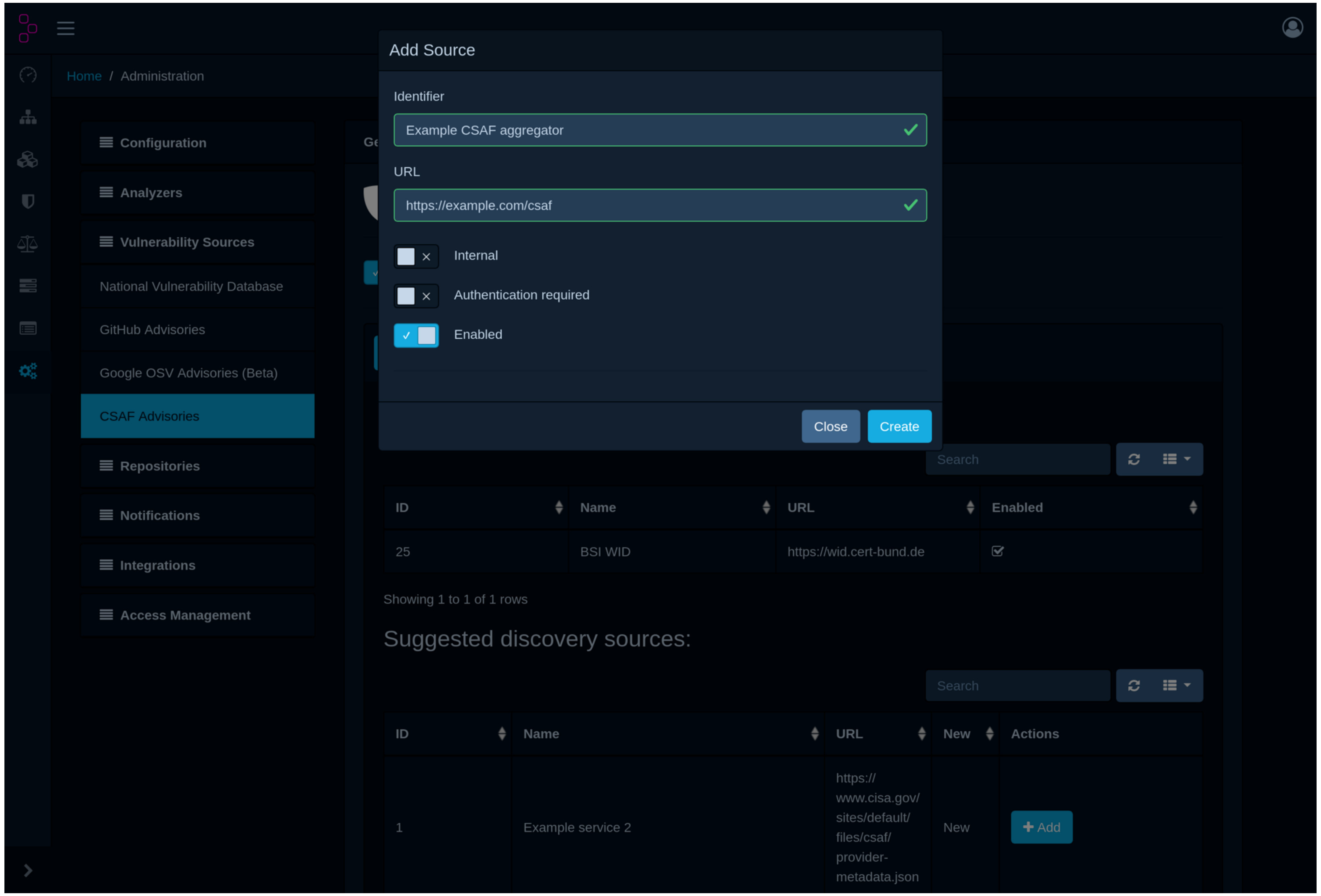
ID	Name	URL	Enabled
25	BSI WID	https://wid.cert-bund.de	✓

Showing 1 to 1 of 1 rows

Suggested discovery sources:

ID	Name	URL	New	Actions
1	Example service 2	https://www.cisa.gov/sites/default/files/csaf/provider-metadata.json	New	+ Add

Add Source: Aggregator



Home / Administration

Configuration
Analyzers
Vulnerability Sources
National Vulnerability Database
GitHub Advisories
Google OSV Advisories (Beta)
CSAF Advisories
Repositories
Notifications
Integrations
Access Management

Add Source

Identifier: Example CSAF aggregator

URL: https://example.com/csaf

- [ ] Internal
- [ ] Authentication required
- [x] Enabled

Close | Create

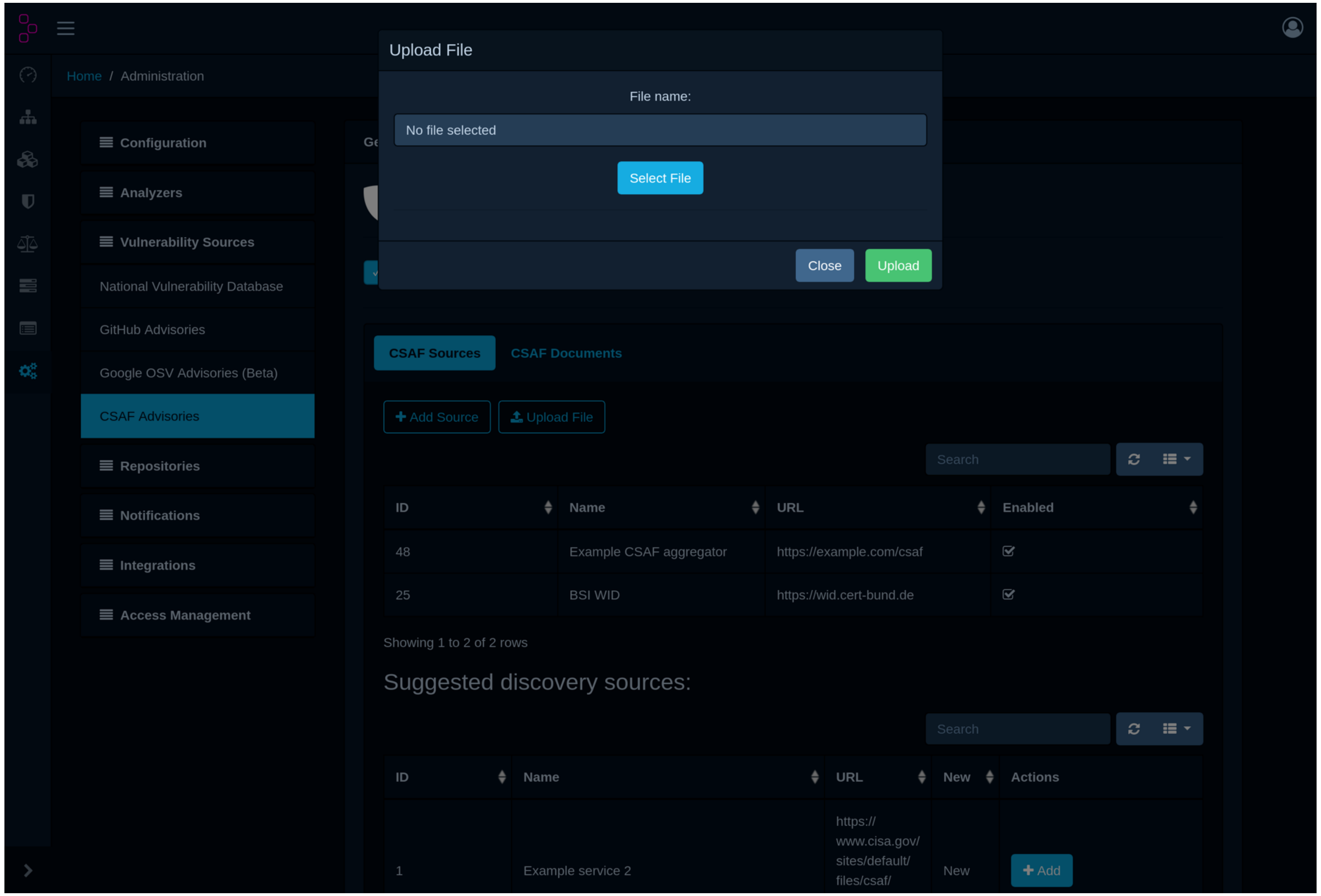
ID	Name	URL	Enabled
25	BSI WID	https://wid.cert-bund.de	✓

Showing 1 to 1 of 1 rows

Suggested discovery sources:

ID	Name	URL	New	Actions
1	Example service 2	https://www.cisa.gov/sites/default/files/csaf/provider-metadata.json	New	+ Add

Add Source: Upload File



Home / Administration

Upload File

File name:

No file selected

ID	Name	URL	Enabled
48	Example CSAF aggregator	https://example.com/csaf	✓
25	BSI WID	https://wid.cert-bund.de	✓

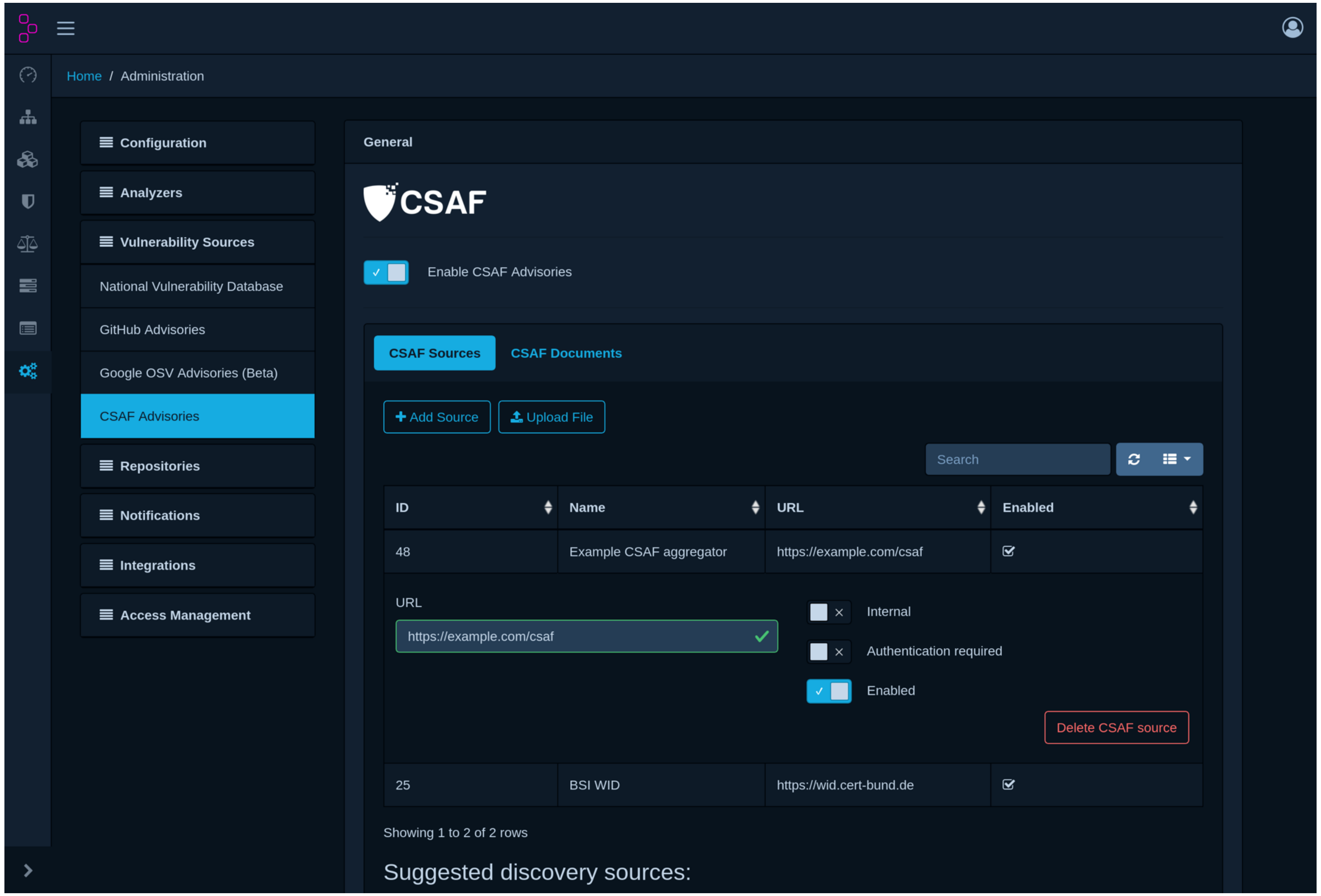
Showing 1 to 2 of 2 rows

Suggested discovery sources:

ID	Name	URL	New	Actions
1	Example service 2	https://www.cisa.gov/sites/default/files/csaf/	New	Add

Fraunhofer AISEC

Edit Sources



Home / Administration

Configuration
Analyzers
Vulnerability Sources
National Vulnerability Database
GitHub Advisories
Google OSV Advisories (Beta)
CSAF Advisories
Repositories
Notifications
Integrations
Access Management

General

CSAF

Enable CSAF Advisories: ✓

CSAF Sources | CSAF Documents

Add Source | Upload File

ID	Name	URL	Enabled
48	Example CSAF aggregator	https://example.com/csaf	✓
25	BSI WID	https://wid.cert-bund.de	✓

URL: https://example.com/csaf

Internal: ✗
Authentication required: ✗
Enabled: ✓

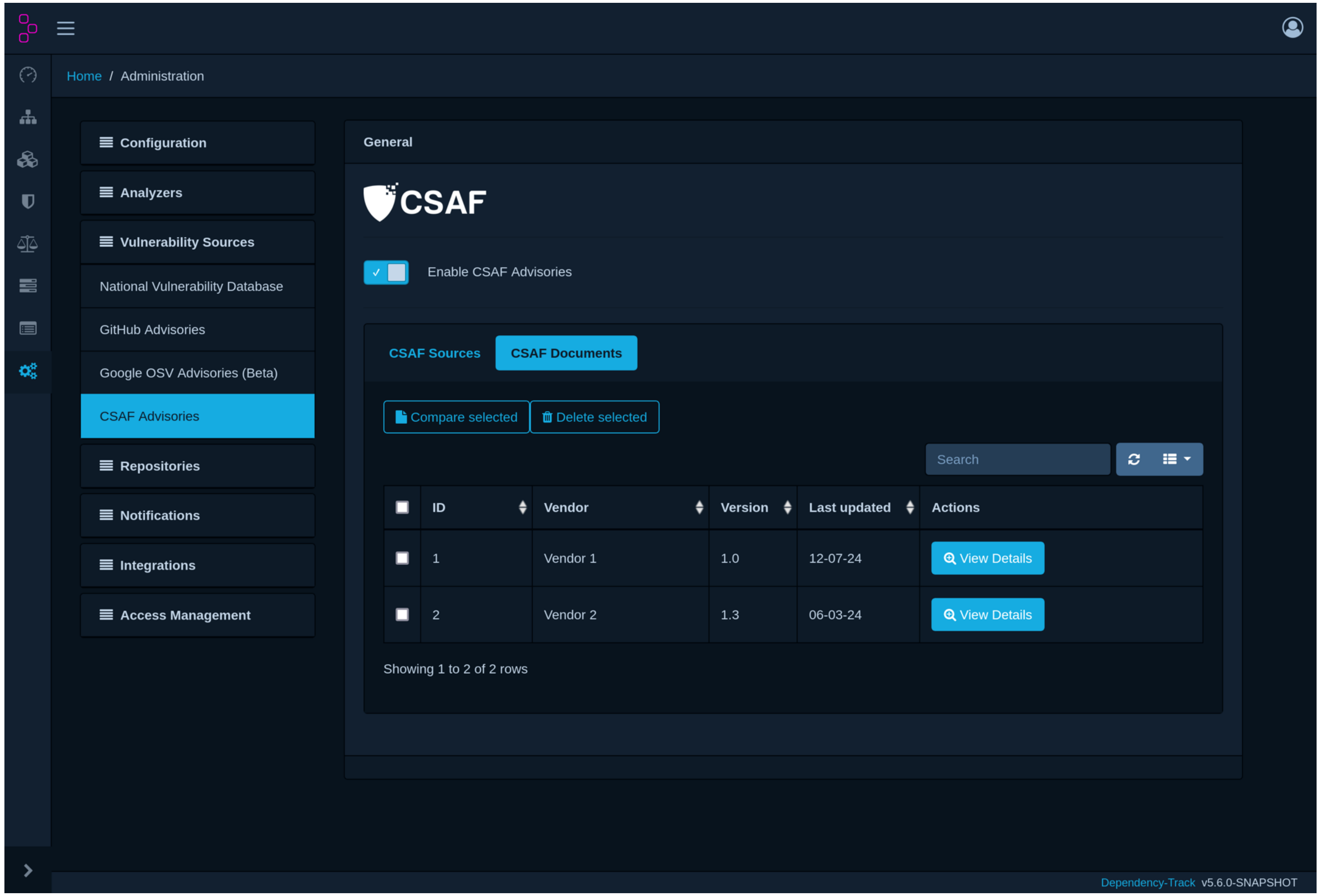
Delete CSAF source

Showing 1 to 2 of 2 rows

Suggested discovery sources:

Fraunhofer AISEC

List documents



Home / Administration

- Configuration
- Analyzers
- Vulnerability Sources
  - National Vulnerability Database
  - GitHub Advisories
  - Google OSV Advisories (Beta)
  - CSAF Advisories
- Repositories
- Notifications
- Integrations
- Access Management

General

CSAF

[✓] Enable CSAF Advisories

CSAF Sources | CSAF Documents

Compare selected | Delete selected

Search

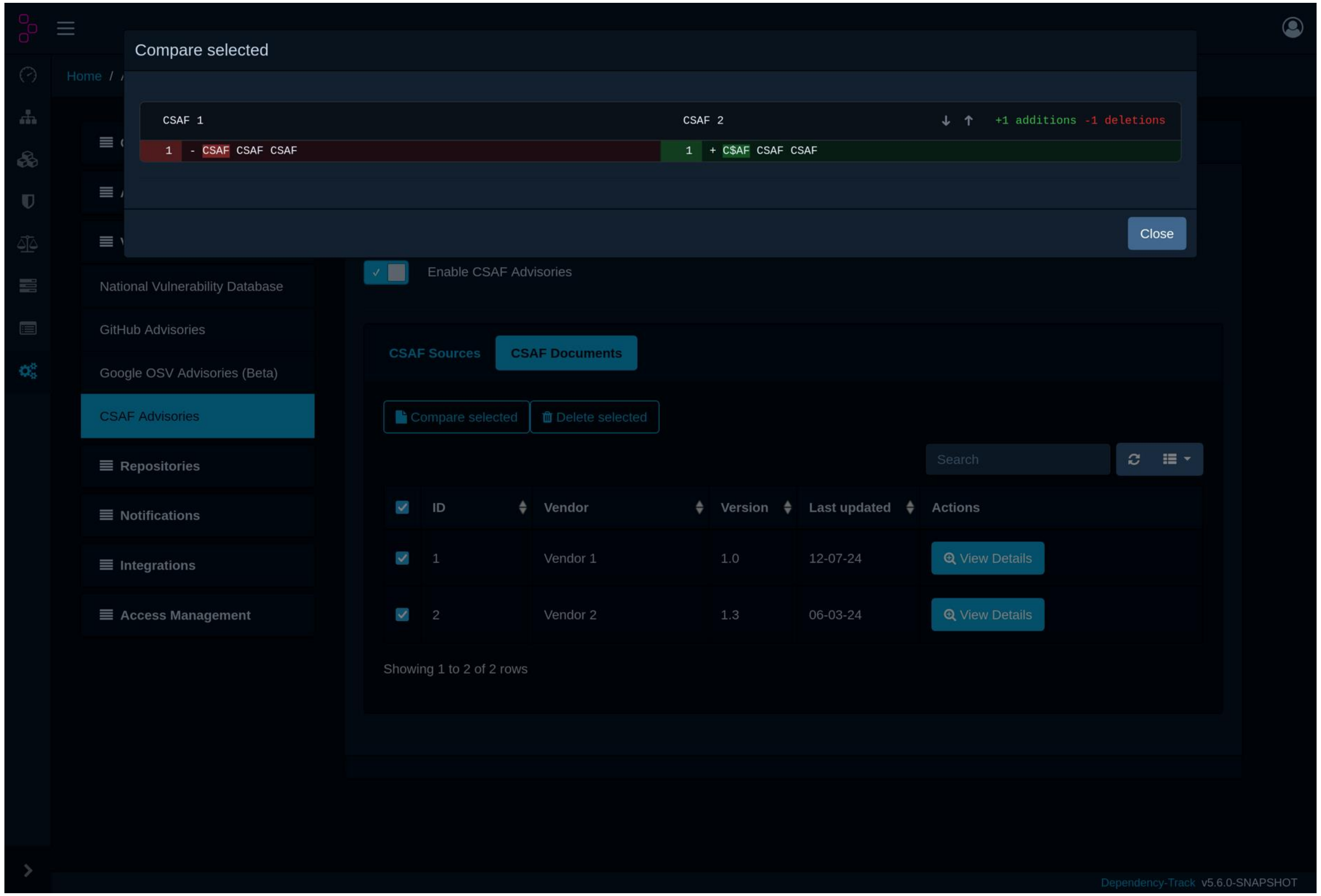
	ID	Vendor	Version	Last updated	Actions
☐	1	Vendor 1	1.0	12-07-24	View Details
☐	2	Vendor 2	1.3	06-03-24	View Details

Showing 1 to 2 of 2 rows

Dependency-Track v5.6.0-SNAPSHOT

Fraunhofer AISEC

Compare Documents



Compare selected

```
CSAF 1                                                    CSAF 2                    +1 additions -1 deletions
1  - CSAF CSAF CSAF                                       1  + C$AF CSAF CSAF
```

Close

Enable CSAF Advisories

	ID	Vendor	Version	Last updated	Actions
✓	1	Vendor 1	1.0	12-07-24	View Details
✓	2	Vendor 2	1.3	06-03-24	View Details

Showing 1 to 2 of 2 rows

Performance

Comparing Performance

Test Case: rhsa-2018_3140.json

Test File:

CSAF document for https://access.redhat.com/errata/RHSA-2018:3140

Why?

It has been mentioned in https://github.com/secvisogram/csaf-validator-service/issues/97 to cause problems in the JavaScript validator

Some Metadata

- 98 MB size
- 31.442 product definitions
- 1.316.188 product references

Running all mandatory tests

6.1.1– 6.1.33

Results

Implementation	Duration
csaf-validator-lib	141s
kotlin-csaf	1s
csaf-rust	not yet (all) implemented

Test 6.1.1

Missing Definition of Product ID

Idea: There should not be a reference to a product ID, which is not defined

Results

Implementation	Duration
csaf-validator-lib	89s
kotlin-csaf	0.73s
csaf-rust	0.41s

Only Schema Validation
+ loading the file

No tests, only loading JSON + schema validation

Results

Implementation	Duration
csaf-validator-lib	89s (???)
go-csaf	2,3s
kotlin-csaf	0.6s
csaf-rust	0.31s

Some thoughts:

- The issue with JS seems to lie in a very inefficient implementation of the schema validation
- Without schema validation, executing the test takes **3s**! still slower than Rust or Kotlin (should still be faster though → inefficient algorithm?)
- Rust and Kotlin do not (yet) return the instance path, but it is expected that this will not slow it down significantly
- Rust and Kotlin are comparable in the speed of the test execution (~100ms); loading + validation is about twice as fast in Rust

Conclusions

Conclusions

kotlin-csaf is ready to use (in your own application)

DependencyTrack integration is progressing well

First PRs to upstream will be done in early 2025

Project will conclude in mid 2025

Rust, Kotlin might be faster than JavaScript (at the very least, the JSON schema validation is WAY faster)

Questions?

Christian Banse
Head of Department „Service & Application Security“
christian.banse@fraunhofer.de

Fraunhofer AISEC
Lichtenbergstraße 11
85748 Garching b. München
www.aisec.fraunhofer.de