Co-funded by the European Union

OCCTET

Open Source Compliance Comprehensive tools and resources designed to simplify and streamline the CRA compliance process for SME, allowing them to tackle the complexities of OSS compliance.

Project Title: Open Source Compliance Comprehensive tools and resources designed to simplify and streamline the CRA compliance process for SME, allowing them to tackle the complexities of OSS compliance.
Project Acronym: OCCTET
Grant Agreement / Contract No.: 101190474
Program: DIGITAL Europe Programme; DIGITAL-ECCC-2024-DEPLOY-CYBER-06
Instrument: DIGITAL JU SME Support Action
Granting Authority: European Cybersecurity Industrial, Technology and Research Competence Centre
Project Start Date: 1 November 2024
Project Duration: 24 months

Deliverable Number: D3.1
Deliverable Title: FedDB-Beta-Del
Deliverable Type (DOA): DEM - Demonstrator, pilot, prototype
Deliverable Type (content): Demonstrator of the federated and shared software metadata platform
Work Package: WP3 - Build Open Source Tools to Automate Evaluation
Task Number(s): T3.2 Reference Data: Deploy Reference Federated and Shared Software Metadata Platform for SBOMs, Code origin and Vulnerabilities
Dissemination Level: PU - Public
Due Date (DoA): 30 April 2025
Actual Submission Date: 30 April 2025
Version: 1.1 (PR1 Revision)

Lead Beneficiary: ABCD- AboutCode
Main Author(s): Philippe Ombredanne - ABCD
Contributing Partner(s): -
Reviewer: ECL - Eclipse Foundation Europe

Document History

Version	Date	Issued By	Status	Comments
0.1	19-03-2025	ABCD	Draft	Initial draft of D3.1
0.2	29-04-2025	ECL	Review	Initial Review
1.0	30-04-2025	ECL	Final	First Release
1.1	18-02-2026	ABCD-ECL	Final	Addressing the recommendations from the Review Report: - Remove repeated example on page 7 - Add an executive summary - Add a conclusion - Use correct document template - Fix minor typos, and formatting

Executive Summary

Deliverable D3.1, "**FedDB-Beta-Del**" is the first demonstrator release for a reference federated software metadata platform under the OCCTET project. It establishes the basis for an open, resilient, and collaborative base system for sharing essential free and open source software (FOSS) metadata (origin, licenses, vulnerabilities) and enables more efficient Cyber Resilience Act (CRA) compliance processes for SMEs.

To overcome the bottlenecks of centralized systems, OCCTET is implementing a decentralized approach building on proven AboutCode open source technologies and designs. This initial demonstrator focuses on providing decentralized access to curated, open-source vulnerability data with key features including

- **Decentralized Data Access**
- **PURL-Based Discovery with AboutCode Hashid**

This demonstrator validates that the core data components are available and can be used, either through direct access keyed by PURL, in bulk or through the public APIs. PURL adoption further validates our approach as the base for the next deliverables in a world where FOSS software packages (and PURL) are the key software supply chain entities.

Keywords: Cyber Resilience Act (CRA), SMEs, FOSS, Open Source Compliance, FederatedCode, VulnerableCode, Package URL (PURL), Software Metadata, Decentralized Data, SBOM.

Table of contents

1 Introduction

1.1 Objectives

- Develop an open-source toolkit to support the efficient creation of Software Bill of Materials (SBOMs), enabling accurate identification and cataloging of software components within digital assets.
- Implement automated vulnerability assessment tools to quickly detect and prioritize software vulnerabilities based on severity and potential impact.
- Support effective remediation by providing actionable insights and recommendations, including patching, version updates, and configuration changes.
- Evaluate the security posture of open-source components, reviewing their maintenance, exposure to known vulnerabilities, and adherence to secure development practices.
- Automate compliance reporting by generating key documents such as SBOMs and Vulnerability Exploitability Exchange (VEX) reports, supporting alignment with Critical Risk Assessment (CRA) standards and easing regulatory compliance.

1.2 Task Overview

Ref task: T3.2 Reference Data: Deploy Reference Federated and Shared Software Metadata Platform for SBOMs, Code origin and Vulnerabilities

A Software Bill of Materials (SBOM) provides a detailed list of software packages. In this task, the goal is to enrich this list with comprehensive metadata. This includes accurate origin and license information, as well as critical data on known security vulnerabilities and available fixes. This enrichment is essential for reducing the compliance burden on small and medium-sized enterprises (SMEs), allowing them to efficiently reuse verified data from multiple sources to support and automate compliance processes.

This task involves deploying a platform that systematically collects, aggregates, and correlates metadata related to open source software (FOSS) packages. This includes information about package origins, known vulnerabilities, and results from secondary analyses such as file inventories, license data, quality assessments, security audits, risk evaluations, and sustainability scores.

The platform will aggregate vulnerability data from major data sources such as the National Vulnerability Database (NVD), GitHub, GitLab, Google OSV, as well as a range of upstream open data sources including Linux distributions and project-specific advisories (e.g., Apache Tomcat, nginx, GNU libc). This information will be correlated for further review and semi-automated curation.

Additionally, the platform will include data to support vulnerability triage, enabling users to assess risks based on factors such as exploitability, reachability, and the presence of vulnerable code. It will be designed as a decentralized and federated system to ensure open reference data can be easily shared among users and stakeholders.

The platform builds upon existing open source projects such as VulnerableCode, PurlDB, and FederatedCode, previously supported by the NGI program. Its goal is to facilitate decentralized data generation (e.g., through package scanning and vulnerability mining), peer review, and collaborative curation. This will be enabled via open and federated technologies such as ActivityPub. In particular, the platform will promote transparent, community-driven peer review of reported vulnerabilities and the open exchange of remediation strategies, supporting collaboration among SMEs and other contributors without reliance on any single vendor, foreign company, or government authority.

2 Federated and Decentralized Metadata System - Introduction to FederatedCode

Managing software metadata - like licenses, origins, or security vulnerabilities - is critical but often inefficient. Most current systems collect and store this data in large, centralized databases. While convenient, this leads to problems like limited access, duplication of effort, and lack of privacy or control.

This approach proposes a federated and decentralized model to manage metadata more openly, collaboratively, and efficiently. A working reference implementation of this model is FederatedCode:

- https://federatedcode.readthedocs.io/
- GitHub: https://github.com/aboutcode-org/federatedcode

2.1 The Problem with Centralized Metadata

Central systems gather scattered metadata into a single database. These offer one-stop access but create new bottlenecks as witnessed in April 2025 with funding issues experienced by the CVE.org centralized vulnerability database program.

- Control issues: One organization decides what data is stored and how. For CVE.org it has been the US MITRE corporation, funded by the US Federal Government
- Data silos: The same data is stored in different places, in different formats. Each database stores data in its own format such as CSAF, OSV, CVE, and for SBOMs using CycloneDX and SPDX.
- Limited sharing: It's hard to share only specific parts of a database. The database is either wholly shared or not at all. Often it grows too big to share and stays locked in a centralized place.
  Redundant effort: Different teams keep repeating the same work - scanning the same software, reviewing the same data over and over.
- Single points of failure: If the central server goes down or is shut off, access is lost. Same if the central organization loses its funding.

2.2 The New Approach: Federated and Decentralized

Our approach suggests using multiple connected data stores where metadata can be published, updated, reviewed, and shared in a decentralized way.
Each piece of metadata, such as info about a software package origin, license, dependencies and vulnerabilities, lives in its own version-controlled repository (e.g., using multiple Git repositories), and updates are shared using open communication protocols either using simple content-addressable direct file download and advertising using protocols like ActivityPub (https://www.w3.org/TR/activitypub/), originally designed for social media.

2.3 Key Components

1. Decentralized Metadata Stores

Software package metadata is stored in many Git repositories as plain text. This includes SBOMs, vulnerabilities, scan results, and more as the usage and need grows. These repositories are distributed, publicly accessible and version-controlled.

The core feature is that metadata is directly discoverable using the identifier of a package, e.g., PURL or Package-URL as specified at https://github.com/package-url/purl-spec
With this identifier, a user can directly and efficiently access and retrieve the metadata of a single package.

2. Federated Updates via ActivityPub

We further plan to deploy a model where each package has its own "account" on a federated ActivityPub network (like the fediverse). Users and machines can then follow packages and get updates when:

- New vulnerabilities are found
- New scan results are added
- License data is corrected or updated

These updates contain pointers back to the Decentralized Metadata Stores

3. Local, Operational Data Stores and Contributions

Each actor in this system can:

- Consume metadata individually, for one package, and/or
- Clone and track one or many Git repositories, and/or
- Subscribe to and participate in ActivityPub data exchanges and conversations, and/or
- Deploy a local operational data store that consists of AboutCode's VulnerableCode and PurlDB, and/or
- Run and contributes scans and metadata enhancements

4. User Involvement

- Users and organizations can:
- Subscribe to only the data they care about
- Contribute their own scans and metadata
- Review and discuss metadata updates
- Choose to trust updates from known contributors

2.4 How It Works in Practice (Examples)

Example 1: Tracking Licenses
A development team using open source packages can check metadata from the federated system. They look up a package by its URL, find license info verified by trusted peers, and get notified if that info changes. If no metadata exists yet, they can contribute their own scans and reviews.

Example 2: Watching for Security Issues
A security team can follow the packages their systems and apps depend on. When a new

vulnerability is published or a fix is released, they get notified. They can also contribute their own insights such as mitigation steps or verification of affected versions back to the system for others to use.

Example 3: Getting reference SBOMs
Modern applications are assembled with 1000's of open source components. Each component can have an SBOM. The distributed approach means that a software or security can fetch an SBOM directly from the shared, open data store and does not need to recompute, recreate and revalidate the SBOM for the open source components.

2.5 Benefits of the Federated Model

- Resilient: No single point of failure.
- Transparent: Full history of updates and who made them.
- Selective: Subscribe to only what you need.
- Collaborative: Share and review metadata like a social network.
- Open and free: Anyone can join and contribute using FederatedCode (https://github.com/aboutcode-org/federatedcode).

2.6 Steps in Deployment

The approach is being deployed in steps:
- First, vulnerability data
- Second, package metadata
- Next, scans and SBOMs

3 Initial FederatedCode Demonstrator

This initial demonstrator focuses on software package vulnerabilities and includes the following key components:

3.1 VulnerableCode

VulnerableCode is an aggregated open-source database that tracks known security vulnerabilities and is continuously updated. It serves as a comprehensive resource for identifying vulnerabilities in software packages.

VulnerableCode is an open-source project that collects, curates, and provides access to publicly known software vulnerabilities mapped to real-world software packages. It helps developers, security teams, and researchers identify which specific packages and versions are affected by known vulnerabilities, using standardized identifiers like PURL as well as CVEs.

Built around the Package URL (purl) specification, VulnerableCode enables precise matching of vulnerabilities to software packages across ecosystems like npm, PyPI, Maven, and more. AboutCode's PURL plays an important role in this supply chain security solution, and has been adopted globally in software bill of materials (SBOM) analysis, and vulnerability management.

Key VulnerableCode resources for this demonstrator include:

- The code is available at: https://github.com/aboutcode-org/vulnerablecode
- The demonstrator is publicly accessible at: https://public.vulnerablecode.io/
- Documentation can be found at: https://vulnerablecode.readthedocs.io

3.2 Decentralized Data Repository

The data used in this demonstrator is published in a Git repository, accessible at:

- https://github.com/aboutcode-data/vulnerablecode-data

3.3 AboutCode Hashid

This data is published using the **AboutCode hashid** library, which can be found at:

- https://github.com/aboutcode-org/vulnerablecode/tree/main/aboutcode/hashid
- The package is also available on PyPI: https://pypi.org/project/aboutcode.hashid/
- The package is also available on PyPI: aboutcode.hashid on PyPI

This library enables the direct access from a PURL to its data using content-defined, hash-based paths to store Vulnerability and Package data using these paths in many balanced directories. The reason why this is needed is to store many vulnerability and

package metadata files, we need to distribute these files in multiple directories and avoid too many files in the same directory which makes every filesystem performance suffer. In addition, when storing these files in Git repositories, we need to avoid creating any repository with too many files that would make using this repository impractical or exceed the limits of some repository hosting services. Therefore we are storing vulnerability data using a directory tree using the first few characters of the PURL hash of a package or the UUID of a vulnerability id.

For example:

```
Python
$ pip install aboutcode.hashid
$ python
>>> from aboutcode import hashid
>>>
hashid.get_package_vulnerabilities_yml_file_path("pkg:maven/com.helger.commons/ph-json")
PosixPath('aboutcode-packages-maven-74/maven/com.helger.commons/ph-json/vulnerabilities.yml')
```

This i"path" s then used to compute the direct download URL at:

https://github.com/aboutcode-data/vulnerablecode-data/blob/main/aboutcode-packages-maven-74/maven/com.helger.commons/ph-json/vulnerabilities.yml

The document at this URL lists all known security vulnerabilities for the "**pkg:maven/com.helger.commons/ph-json**" PURL. A given vulnerability can then be accessed by its URL as with:

https://github.com/aboutcode-data/vulnerablecode-data/blob/main/aboutcode-vulnerabilities/vd/VCID-vdum-vhwp-aaag.yml

3.4 Other components for the solution

Other components that will be part of the final solution are:

3.4.1 FederatedCode

The code federation layer. This is available and publicly deployed, but not yet operational in this version of the demonstrator.

- Code at https://github.com/aboutcode-org/federatedcode
- Docs at https://federatedcode.readthedocs.io
- Demo system at https://demo.data.aboutcode.org/

3.4.2 PurlDB

The database of all package metadata keyed by PURL. This is available and publicly deployed, but not yet operational in this version of the demonstrator.

PurlDB (Package URL Database) is a structured, open database designed to store and manage metadata about software packages across various ecosystems using the Package URL (purl) specification. It provides a unified way to represent package coordinates and their related information such as licenses, versions, and repository metadata across diverse package managers like npm, PyPI, Maven, and others.

PurlDB is designed for use in software composition analysis (SCA), supply chain security, and vulnerability tracking tools. It aims to make software package data accessible, consistent, and easily queryable for developers, security analysts, and automated tools.

- Code at https://github.com/aboutcode-org/purldb
- Docs at https://aboutcode.readthedocs.io/projects/PURLdb
- Demo system at https://public.purldb.io/api/

4 Conclusion

This Deliverable D3.1, "**FedDB-Beta-Del**" is the first released demonstrator for the reference federated software metadata platform as part of the OCCTET project. This demonstrator and the corresponding open source code and data does establish the basis for an open, decentralized and distributed system for sharing and accessing essential free and open source software (FOSS) metadata (code origin, licenses, vulnerabilities, and SBOMs) to enable automation for software composition analysis and CRA compliance processes for all software teams, with a specific support for SMEs that do not have the resources to compute, revalidate or acquire that data.

This demonstrator validates that the core code and data components are available and can be used, either through direct access keyed by PURL, from data repositories cloning, or with the REST JSON VulnerableCode API.

Future and ongoing efforts will focus on extending this demonstrator and supporting the work on the BasicChain deliverables. In addition we are planning for a significant refactoring of the API and data model away from a vulnerabilities-centric approach, shifting towards a data and API model centered on software packages and software package vulnerability advisories. Our approach to provide open data keyed by PURL has been validated and reinforced by the adoption of PURL throughout the industry for SCA tools, CRA compliance tools, Vulnerability management, Vulnerability databases, SBOMs and VEXs schema standards, including the CVE schema and its upcoming standardization at Ecma and at ISO.

5 ACRONYMS AND ABBREVIATION

CRA - Cyber Resilience Act
ENISA - European Union Agency for Cybersecurity
EU - European Union
GDPR - General Data Protection Regulation
FOSS - Free and Open-Source Software
AI - Artificial Intelligence
IPR - Intellectual Property Rights
KPI - Key Performance Indicator
PII - Personally Identifiable Information
RBAC - Role-Based Access Control
SBOM - Software Bill of Materials
SAST - Static Application Security Testing
DAST - Dynamic Application Security Testing
MFA - Multi-Factor Authentication
SOC - Security Operations Centre
IR - Incident Response
API - Application Programming Interface
TLS - Transport Layer Security
ISO - International Organization for Standardization
IEC - International Electrotechnical Commission
ETSI - European Telecommunications Standards Institute
SUS - System Usability Scale
TRL - Technology Readiness Level
WP - Work Package
DoA - Description of Action

6 BIBLIOGRAPHY

1. **European Commission**, The Cyber Resilience Act - Questions & Answers, European Commission, 2024.
2. **European Commission**, DIGITAL Europe Programme - Model Grant Agreement, European Commission, 2024.
3. **OCCTET Project Consortium**, Description of Action (DoA), Grant Agreement No. 101190474, 2024.
4. **OCCTET Project Consortium**, CRA SME requirements and self-assessment checklists (D1.2), 2025.