

OWASP DEPENDENCY-TRACK

Community Meeting
October 2025

Organizational

- Community meetings are recorded and uploaded to [YouTube](#)
- Slides will be published in the [DependencyTrack/community](#) repository
- Please use the Zoom chat to ask questions during the presentation
- There will be an open Q&A section towards the end

Call for Guest Presentations

- Want to brag about the cool DT setup you've built?
- Want to vent about what needs improvement?
- Want to get input on DT-related designs?
- Want to propose changes?

We'd love to host you here!



Agenda

1. Upcoming v4.13.5 Release
2. Important Notices
3. Sneak Peek: v5 Plugin System
4. Q&A

Upcoming Releases

Upcoming Releases: v4.13.5

- Fixes BOM validation failures due to unknown SPDX license IDs
 - Particularly SMAIL-GPL has been causing issues recently
- Fixes CPE matching being case-sensitive
- Fixes medium severity vulnerability
 - Details to be disclosed upon release
- Requires authentication for OSS Index analyzer
- Aiming for release this week
- Release by Monday at the latest

Important Notices

NVD JSON 1.0 Data Feeds Discontinuation

- If:
 - You mirror the NVD via feed files (default)
 - You're still on Dependency-Track version <v4.13.4
- **PLEASE upgrade to >=v4.13.4 ASAP**
 - Or switch to API-based mirroring in the meantime
- You will not get CVE updates otherwise

Configuration

Analyzers

Vulnerability Sources

National Vulnerability Database

GitHub Advisories

Google OSV Advisories (Beta)

Repositories

Notifications

Integrations

Access Management

General

Enable National Vulnerability Database mirroring

The National Vulnerability Database (NVD) is the largest publicly available source of vulnerability intelligence. It is maintained by a group within the National Institute of Standards and Technology (NIST) and builds upon the work of MITRE and others. Vulnerabilities in the NVD are called Common Vulnerabilities and Exposures (CVE). There are over 100,000 CVEs documented in the NVD spanning from the 1990's to the present.

This product uses data from the NVD API but is not endorsed or certified by the NVD.

NVD Feeds URL

`https://nvd.nist.gov/feeds`



Enable mirroring via API

[Why should I enable API mirroring?](#)

Additionally download feeds

[Feeds will not be parsed, but made available to other clients at `/mirror/nvd`](#)

API endpoint

`https://services.nvd.nist.gov/rest/json/cves/2.0`

API key

.....

[How do I get an API key?](#)

Last Modification (UTC)

No date selected

No time selected

[After mirroring the NVD database once completely, all following mirror operations will only request data that was modified since its last successful execution.](#)

[Changing the last modification datetime manually is generally not recommended, but may be used to force re-ingestion of NVD data. Note that due to a limitation in the NVD's REST API, only data for 120 consecutive days can be requested when a last modification datetime is configured. Resetting the last modification datetime will cause the entire NVD database to be re-mirrored.](#)

Update

OSS Index Authentication Requirement

Authentication Required

OSS Index now requires authentication. Learn why this improves stability, how to set up your token, and how upcoming paid tiers enable unlimited access.

[Create account](#)

[What is an API token](#)

Sonatype OSS Index now requires all users — including automated tools — to authenticate using a personal [API token](#).

What's changing:

- All Sonatype OSS Index web and API access now requires authentication.
- Anonymous requests are no longer allowed.
- Most integrations will start failing if a token is not configured.

Why this is happening:

- High anonymous traffic (especially from automated tools) has made it harder to maintain stable, fair service.
- Authentication allows us to give developers more control and avoid one-size-fits-all limits.
- It also sets the foundation for usage-based tiers and future product improvements.

What you get when you authenticate:

- Higher rate limits, tied to your usage — not your shared IP.
- Better reliability, with traffic shaping based on real users.
- Tool-specific setup instructions, pre-filled with your token.
- Future usage visibility, support access, and optional upgrades.

How to get started:

1. [Create an account](#) — it's free
2. Get your [API token](#)
3. Configure your tools

<https://ossindex.sonatype.org/doc/auth-required>

OSS Index Authentication Requirement

- If:
 - You use Sonatype OSS Index for vulnerability analysis (default)
 - You have not configured credentials for it in the settings
- **PLEASE** configure credentials **ASAP**
- Unauthenticated access no longer possible since ~last weekend

Sneak Peek: v5 Plugin System

Plugin System - Why?

- Byproduct of our own internal needs
- Driven by many cases of:
 - Feature **X** backed by N possible implementations
 - Expectation of N growing over time
 - Primarily integrations with 3rd party services
 - Implementations needing to be configurable by users
 - Implementations being too reliant on platform internals
 - Repetitive lifecycle / resource management
- But also:
 - Need to modularize the codebase more aggressively
 - Intent to enable community to implement custom integrations
- Inspired by other extensible systems (Keycloak, Trino, ...)

Plugin System - Concepts

- Dependency-Track defines *extension points*
- Zero or more *extensions* can implement an *extension point*
- *Extension* lifecycle is managed by *extension factories*
- A *plugin* is a collection of one or more *extension factories*
- *Plugins* can be *built-in* or *external* (i.e., .jar files)

Plugin System - Lifecycle

- *Plugins* are discovered on application startup
- *Extension factories* are created and initialized *once*
- *Extension* instances are created ad-hoc when needed
- *Extension factories* and *extensions* release resources after use
 - e.g. connections, temporary files, caches

Plugin System - (Intentional) Limitations

- Dependency-Track controls *when* an extension point is invoked
- Extensions *do not* get full platform access
 - *Can not* interact with database directly
 - *Can not* register API endpoints
 - *Can not* see other plugins or extensions
 - *Can not* see code from the main codebase
- *Not meant to make everything extensible*
 - Only specific aspects of the platform that benefit from it

Plugin System - Configuration

- Differentiation between *deployment* and *runtime* configuration
 - *Deployment* configs are static and do not change at runtime
 - System properties, environment variables, ...
 - *Runtime* configs are dynamic and may change at runtime
 - Database rows, modifiable by users via UI
- Extensions decide *where* to retrieve *which* configuration from
- Extensions can only access their own configuration
- Configuration UI is rendered based on *runtime* config definitions

```
CONFIG_ENABLED = new RuntimeConfigDefinition<Boolean>(
    "enabled",
    "Whether the GitHub Advisories data source should be enabled",
    ConfigTypes.BOOLEAN,
    /* defaultValue */ false,
    /*isRequired */ false,
    /*isSecret */ false);
CONFIG_ALIAS_SYNC_ENABLED = new RuntimeConfigDefinition<Boolean>(
    "alias.sync.enabled",
    "Whether to include alias information in vulnerability data",
    ConfigTypes.BOOLEAN,
    /* defaultValue */ false,
    /*isRequired */ false,
    /*isSecret */ false);
CONFIG_API_URL = new RuntimeConfigDefinition<URL>(
    "api.url",
    "Base URL of GitHub's GraphQL API",
    ConfigTypes.URL,
    /* defaultValue */ defaultApiUrl,
    /*isRequired */ true,
    /*isSecret */ false);
CONFIG_API_TOKEN = new RuntimeConfigDefinition<String>(
    "api.token",
    "Access token to authenticate with the GitHub API",
    ConfigTypes.STRING,
    /* defaultValue */ null,
    /*isRequired */ false,
    /*isSecret */ true);
CONFIG_WATERMARK = new RuntimeConfigDefinition<Instant>(
    "watermark",
    "Highest observed modification timestamp of processed vulnerabilities",
    ConfigTypes.INSTANT,
    /* defaultValue */ null,
    /*isRequired */ false,
    /*isSecret */ false);
```

Secret configs are transparently encrypted.

Example definition of runtime configurations.

Configs are properly typed.

Configuration

Analyzers

Vulnerability Sources

National Vulnerability Database

GitHub Advisories

Google OSV Advisories (Beta)

Repositories

Notifications

Integrations

Access Management

github

enabled
Whether the GitHub Advisories data source should be enabled

alias.sync.enabled
Whether to include alias information in vulnerability data

api.url
 ✓

Base URL of GitHub's GraphQL API

api.token
 ✓

Access token to authenticate with the GitHub API

watermark
 ✓

Highest observed modification timestamp of processed vulnerabilities

Form in the UI, rendered based on runtime config definitions of an extension.

Plugin System - Existing Extension Points

- **File storage**
 - Used to persist files between async processing steps
 - Implemented: Memory, Local Filesystem, S3



ADR-005

File storage plugin

- **Vulnerability data sources**
 - Implemented: NVD, GitHub, OSV

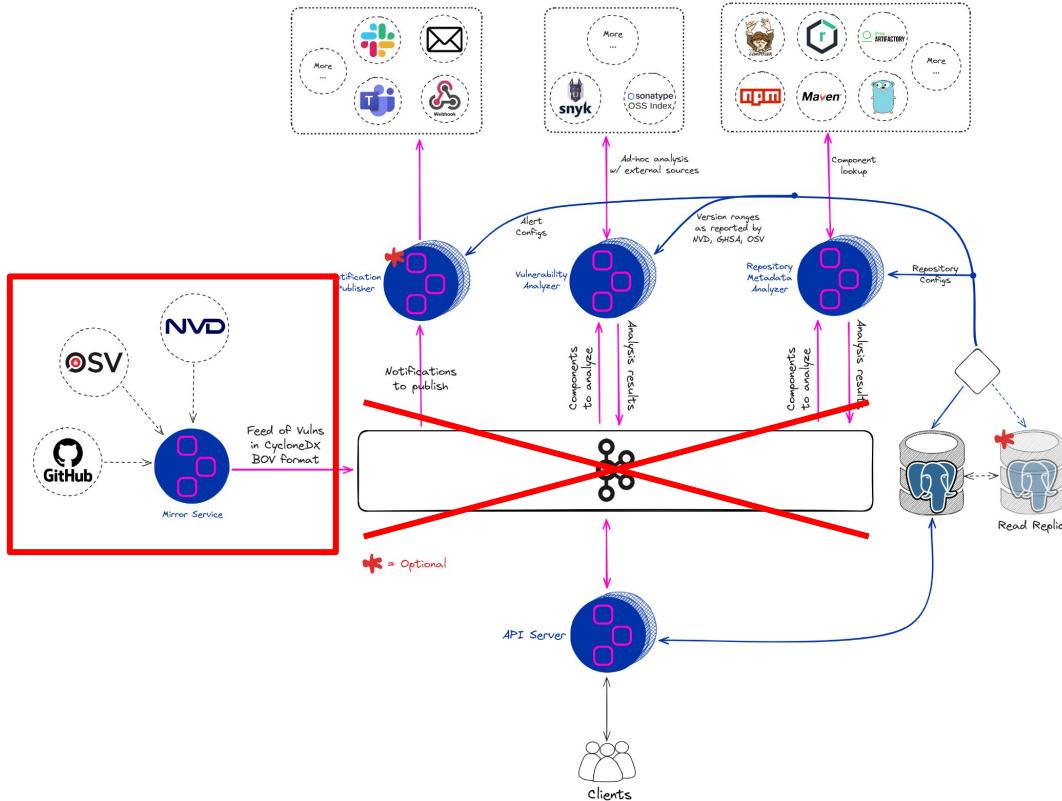


ADR-010

Vulnerability data
source extension point

Plugin System - Example

Extension point is used to get rid of this service.



Plugin System - Example

Retain pull-based semantics of Kafka.

```
/**  
 * A source of vulnerability intelligence data.  
 * <p>  
 * Data is exchanged in the CycloneDX Bill of Vulnerabilities (BOV) format.  
 *  
 * <h3>Expected {@link Iterator} behavior</h3>  
 * It is expected that sources make an effort to keep as little data as possible  
 * in memory, and lazily retrieve new data as {@link Iterator#hasNext()} is invoked.  
 *  
 * Sources may deviate from this behavior to improve performance by buffering,  
 * or to avoid rate limiting of external APIs.  
 *  
 * @see <a href="https://cyclonedx.org/capabilities/bov/">CycloneDX BOV</a>  
 * @since 5.7.0  
 */
```

```
public interface VulnDataSource extends ExtensionPoint, Iterator<Bom> {
```

```
/**  
 * Marks a given BOV as processed, enabling the data source to advance its watermark.  
 * <p>  
 * This is only relevant for data sources that support incremental retrieval.  
 *  
 * @param bov The BOV to be marked as processed.  
 */  
default void markProcessed(final Bom bov) {  
}
```

Avoid redundant processing across invocations.

Definition of the VulnDataSource extension point.

Plugin System - Example

```
Collection<ExtensionFactory<VulnDataSource>> dataSourceFactories =  
    pluginManager.getFactories(VulnDataSource.class);  
  
for (ExtensionFactory<VulnDataSource> dataSourceFactory : dataSourceFactories) {  
    try (VulnDataSource dataSource = dataSourceFactory.create()) {  
        while (dataSource.hasNext()) {  
            Bom bov = dataSource.next();  
            process(bov);  
            dataSource.markProcessed(bov);  
        }  
    }  
}
```

Simplified usage of the VulnDataSource extension point.

Plugin System - Upcoming Extension Points

- Notification publishers
 - Webhook, Microsoft Teams, Slack, Mattermost, ...
- Vulnerability analyzers
 - OSS Index, Trivy, Snyk, ...
- Data sharing(?)
 - Upload / sync with Defect Dojo, ...
- BOM formats(?)
 - SPDX, Syft, ...

Plugin System - TODOs

- Loading of external plugins (with classloader isolation)
- i18n mechanism for configuration UI
- Publishing of plugin API to Maven Central
- Plugin authoring guide

Caveat:

- Making it work for our internal use cases has priority
- Opening it for external plugins is planned but on the backburner

Q&A