OWASP DEPENDENCY-TRACK

Community Meeting
July 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 Releases
2. Crash Course: Access Control
    a. Authentication
        i. User Management
        ii. API Keys
    b. Authorization
    c. Portfolio Access Control
    d. Gaps & Limitations
3. Q&A

Upcoming Releases

Upcoming Releases: v4.13.3

- To be released over the coming days
- ~10 bugfixes that landed since the v4.13.2 release

Upcoming Releases: v4.14.0

- No planned release date yet
- Expected to be the last v4 minor version prior to v5
- Activity slowed down due to increased focus on getting v5 ready
- See glimpse of v5 activities in the May community meeting

New Container Image Variant

- Based on Alpine instead of Debian
- ~200MB smaller than existing image
- Significantly fewer OS package vulns
- Published *in addition* to existing image
- Debian-based image to be discontinued in v5

dependencytrack/apiserver:snapshot
339,5 MB, created 4 hours ago

dependencytrack/apiserver:snapshot-alpine
147,2 MB, created 4 hours ago

New Container Image Variant

```
nscur0@devastation:~
$ trivy image -q --scanners vuln --pkg-types os --table-mode summary dependencytrack/apiserver:snapshot

Report Summary

| Target | Type | Vulnerabilities |
|---|---|---|
| dependencytrack/apiserver:snapshot (debian 12.11) | debian | 81 |

Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)
```

```
nscur0@devastation:~
$ trivy image -q --scanners vuln --pkg-types os --table-mode summary dependencytrack/apiserver:snapshot-alpine

Report Summary

| Target | Type | Vulnerabilities |
|---|---|---|
| dependencytrack/apiserver:snapshot-alpine (alpine 3.22.0) | alpine | 0 |

Legend:
- '-': Not scanned
- '0': Clean (no security findings detected)
```

Crash Course: Access Control

Authentication

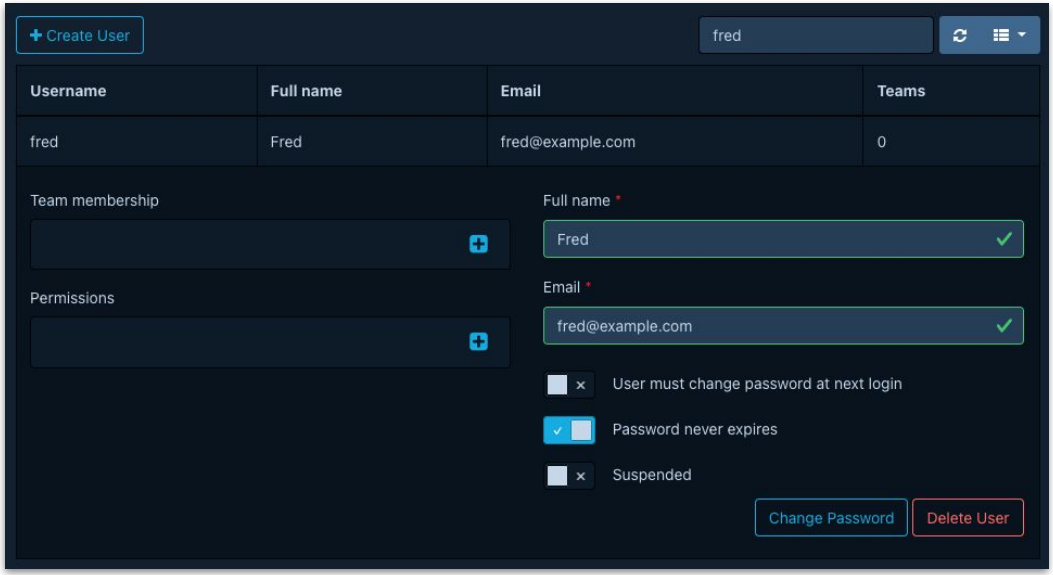
Authentication: Concepts

- Users
  - Actual humans (d'uh) interacting with the application
  - Multiple options of management
- Teams
  - A logical group of users
- Bearer Tokens
  - JWTs issued to users upon successful authentication
  - Frontend takes care of handling them, hidden from end users
- API Keys
  - Credentials to authenticate on behalf of a team

Managed Users

Basic built-in user management

AuthN via username + password



Managed Users: When to Use

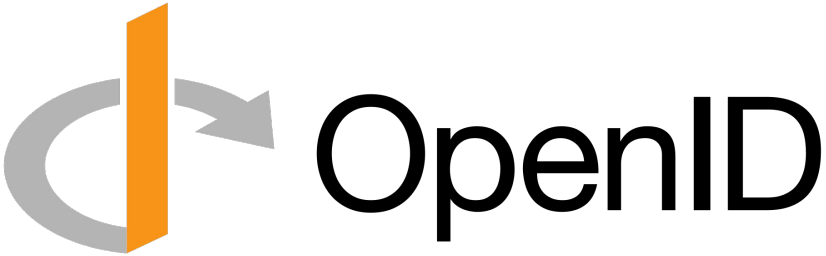
- You’re just fooling around
- You have a small, relatively stable user base
- You want precise control over who can access the system
- You have no central identity management solution available

OpenID Connect

Works with most popular identity providers.

IdP information is synchronized ad-hoc whenever a user logs in.

(Configurable, more on that later)



OpenID Connect

Users are uniquely identified by their subject (`sub`) claim.

(This value is random gibberish most of the time)

User name taken from configurable claim (e.g. `email`).

DT team membership can be assigned based on configurable claim (e.g. `groups`).

```
1  {
2      "sub": "fred",
3      "email": "fred@example.com",
4      "email_verified": true,
5      "groups": [
6          "acme-inc-devs",
7          "acme-inc-devs-integrations"
8      ]
9  }
```

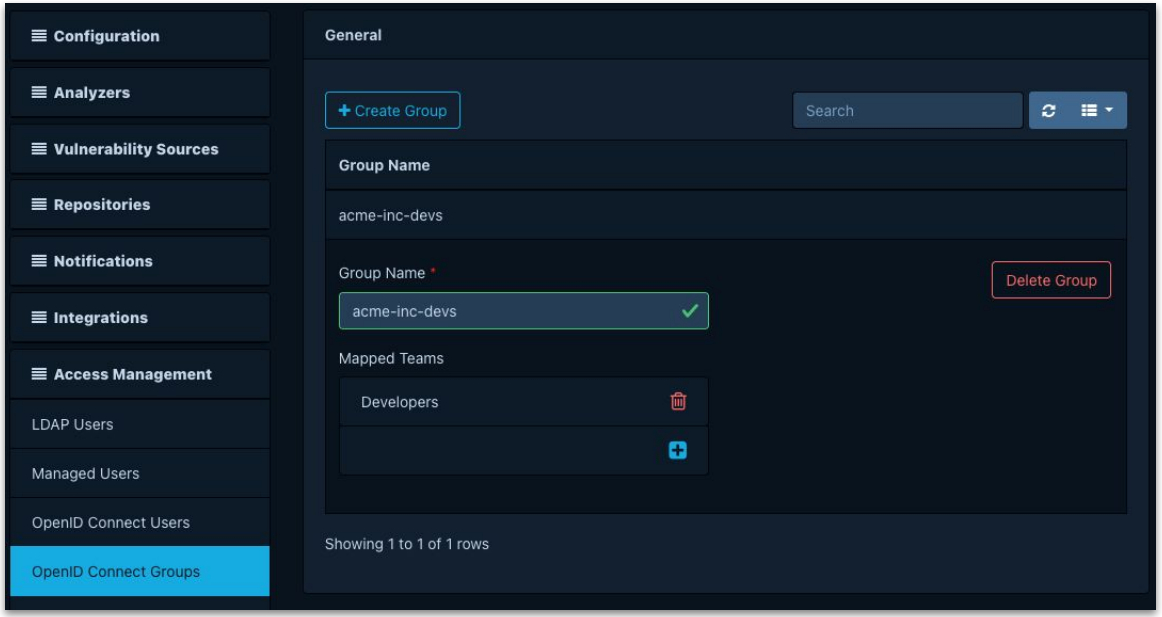
Minimal example claims from an OIDC ID token

OpenID Connect

Mapping to translate from IdP groups to DT team names.

Necessary because OIDC has no group discovery mechanism.

Can act as filter.
(only sync relevant groups)



OpenID Connect: Modes of Operation

Feature Combination	Outcome
`alpine.oidc.user.provisioning=true` `alpine.oidc.team.synchronization=true`	New users auto-provisioned upon login. Team membership automatically synchronized.
`alpine.oidc.user.provisioning=true` `alpine.oidc.team.synchronization=false`	New users auto-provisioned upon login. Manually assigned team memberships.
`alpine.oidc.user.provisioning=false` `alpine.oidc.team.synchronization=true`	Only existing users can authenticate. Team membership automatically synchronized.
`alpine.oidc.user.provisioning=false` `alpine.oidc.team.synchronization=false`	Only existing users can authenticate. Manually assigned team memberships.

OpenID Connect: Docs with Examples

https://docs.dependencytrack.org/getting-started/openidconnect-configuration/

OpenID Connect: When to Use

- You have a medium to large, and / or diverse user base
- A central identity management solution is available
- You use multiple tools and want to offer SSO to users
- You want to enforce 2FA, password policies, etc.

LDAP

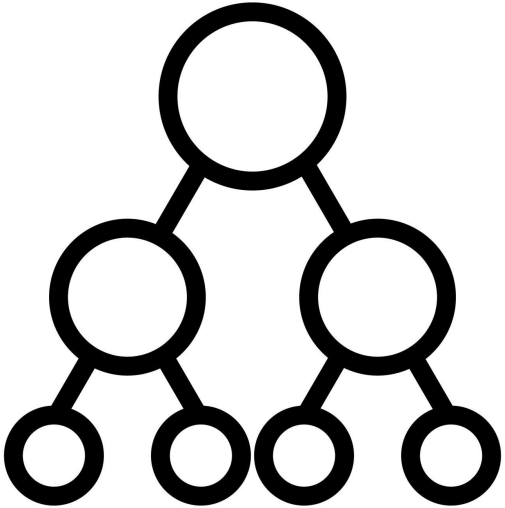
Users are uniquely identified by their distinguished name (DN).

Users and groups synced automatically with directory server.
(And refreshed every 6h)

Configurable username and group filters.

Customizable LDAP search query filters for eligible users and groups.
(You probably don't want to sync your entire org)

Like OIDC, user provisioning and team membership sync is configurable.



LDAP: Docs with Examples

https://docs.dependencytrack.org/getting-started/ldap-configuration/

LDAP: When to Use

- You're heavily invested in LDAP / Active Directory
- OpenID Connect is not available
  - Consider using identity federation (e.g. with Keycloak)

User Management: Recommendations

- Commit to using *one* user management method
- Use OpenID Connect
  - IAM scattered across many services is a pain, don't do it
  - As far as we know more widely adopted by community than LDAP
- Enable auto-provisioning of new users, *and* team synchronization
  - Initial setup overhead, but pretty much hands-off from then on
- Keep the managed `admin` user around for emergency access
  - If the IdP is down or the integration breaks, you're not locked out
  - Goes without saying you should assign a strong password

API Keys

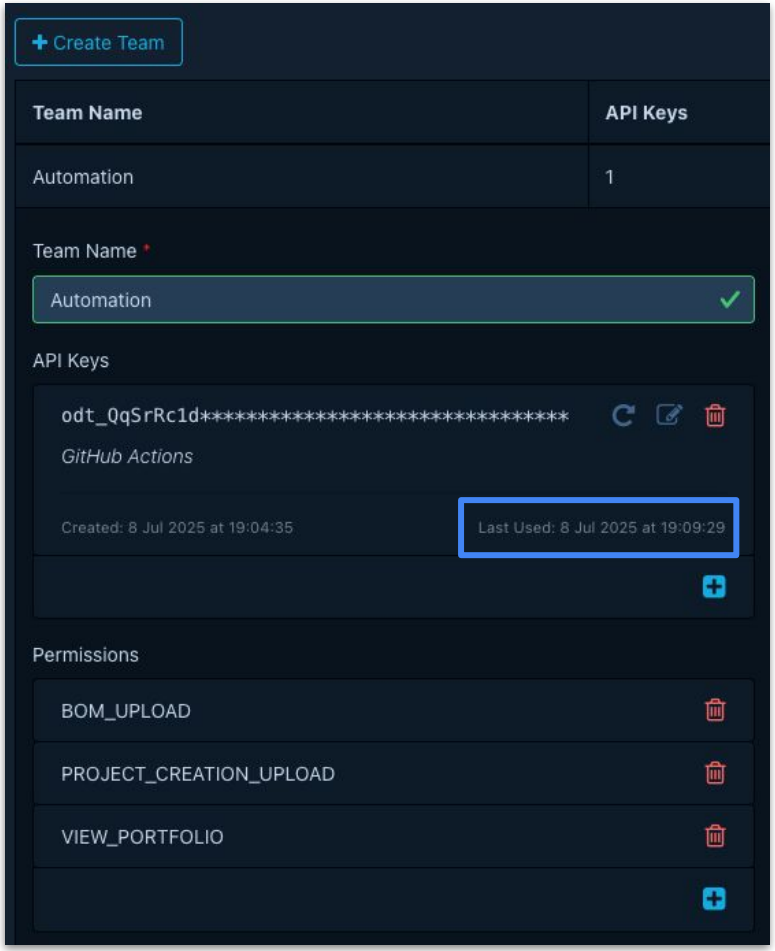
Format:
`odt_<publicId>_<key>`

Created at the team-level.

Inherit permissions of the team.

Comments to describe what a key is for.

Most recent usage is tracked.



API Keys

To be provided via the `X-Api-Key` header.

```
nscur0@devastation:~
$ curl -s \
-H 'X-Api-Key: odt_QqSrRc1d_Pkt9ALOsqgktHQ0rZU83TF1Px11Qw6n6' \
http://localhost:8080/api/v1/team/self | jq .
{
  "uuid": "92144d4f-fe8e-4363-a213-6d726972cb29",
  "name": "Automation",
  "permissions": [
    {
      "name": "BOM_UPLOAD",
      "description": "Allows the ability to upload CycloneDX Software Bill of Materials (SBOM)"
    },
    {
      "name": "PROJECT_CREATION_UPLOAD",
      "description": "Provides the ability to optionally create project (if non-existent) on BOM or scan upload"
    },
    {
      "name": "VIEW_PORTFOLIO",
      "description": "Provides the ability to view the portfolio of projects, components, and licenses"
    }
  ]
}
nscur0@devastation:~
$
```

API Keys: When to Use

- *Any* automation use-case, including:
- CI / CD pipelines
- 3rd party integrations
- Custom scripts

API Keys: Recommendations

- Create dedicated *teams* for your automation use-cases
- Assign a comment to each key so you know what they are for, or who they are used by
- Regularly check for API keys that haven't been used in a long time, and remove those
- Use tooling like *trufflehog* or *Gitleaks* to spot leaked keys in your VCS
  - Might require custom rules ATM, we should consider contributing some :)

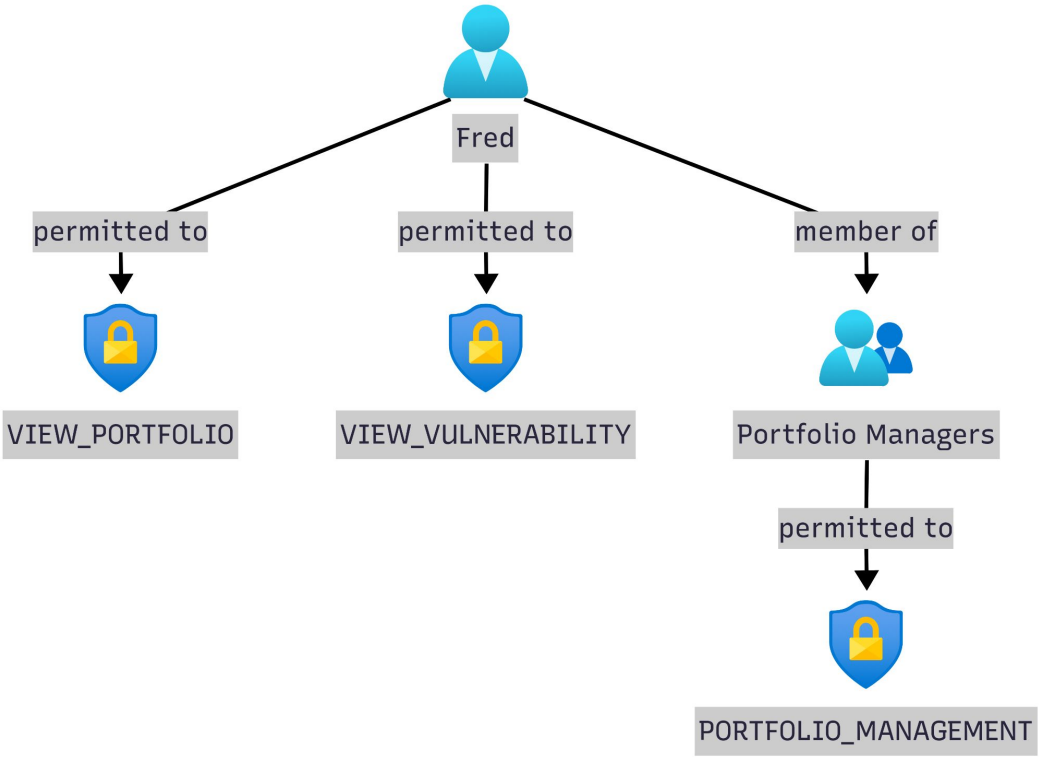
Authorization

Authorization: Concepts

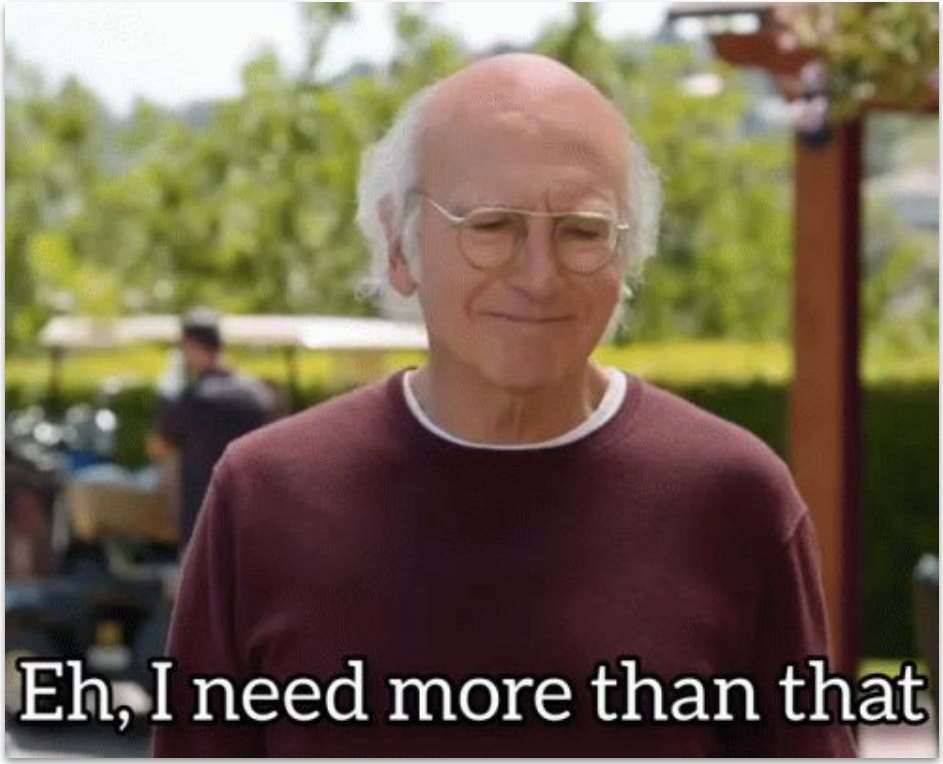
- Users have permissions
- Teams have permissions
- Users can be members of 0-N teams
- Effective Permissions of a user:

  `permissionsOf(user) + permissionsOf(user.teams)`
- Permissions apply to *all* projects
  - Depending on your org structure and user base, this can be totally sufficient

Authorization: Effective User Permissions



Authorization



- You, probably.

Portfolio
Access Control

Portfolio Access Control

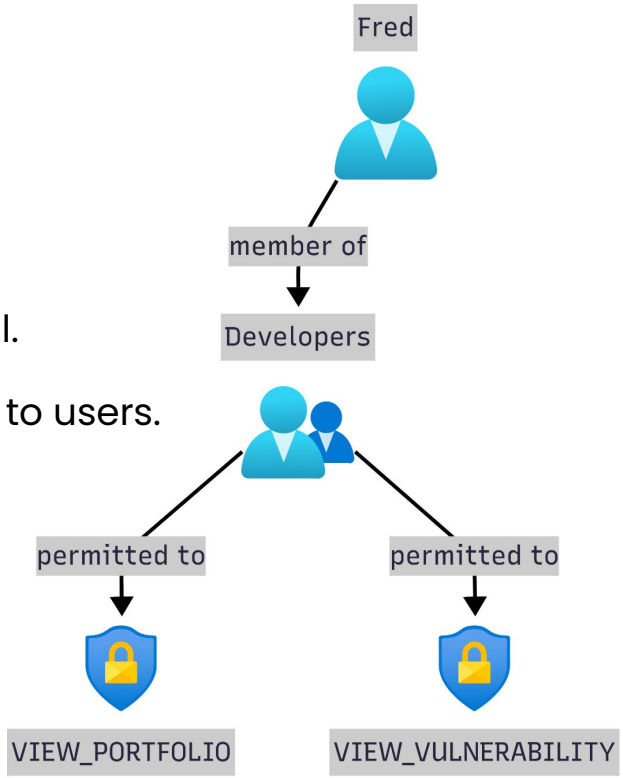
- Opt-in (technically still beta status)
- Controls which projects are accessible *by which teams*
- Permissions still apply to *all* accessible teams
- Bypassed by users with `ACCESS_MANAGEMENT` permission
  - Anyone who has this is effectively administrator
  - Only `admin` user has it by default

Portfolio Access Control: Example Setup

Separate teams for *permissions* and *project access.*

Common team for permissions, akin to Role Based Access Control.

No permissions assigned directly to users.

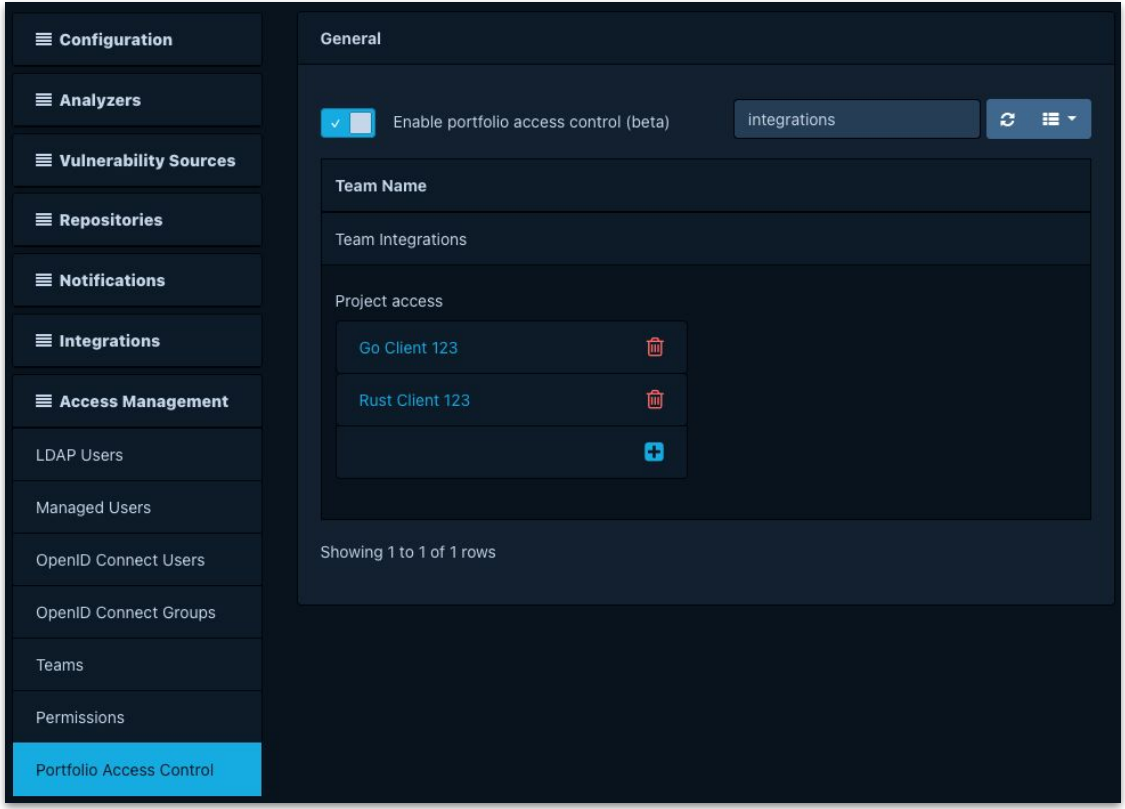


Portfolio Access Control: Assigning Access

- Manually by administrators via UI
- Manually by team members on project creation
- Implicit via BOM upload
- Automated via REST API

Portfolio Access Control: Assigning Access

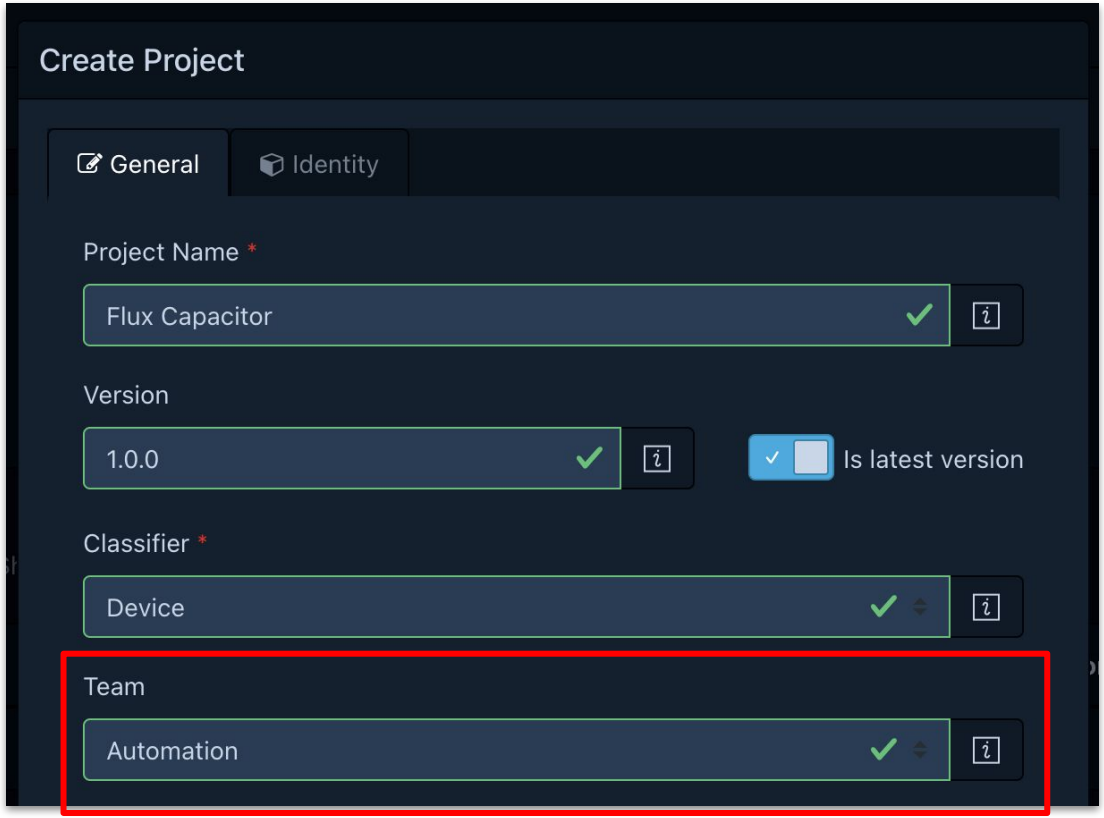
Admins can assign or remove access for all teams in the administration panel.



Portfolio Access Control: Assigning Access

When creating a new project, users can select one of the teams they are a member of.

Admin users can select *any* team here.



Portfolio Access Control: Assigning Access

Uploading a BOM to a *new project* via REST API will assign the team owning the API key to the project.

The team must have the `PROJECT_CREATION_UPLOAD` permission.

Request parameter `autoCreate` must be `true`.

Existing projects remain unmodified.

nscur0@devastation:~

```
nscur0@devastation:~
$ curl -v -X POST \
-H 'Content-Type: multipart/form-data' \
-H 'X-Api-Key: odt_QqSrRc1d_Pkt9ALOsqgktHQ0rZU83TF1Px11Qw6n6' \
-F 'projectName=Foo' \
-F 'projectVersion=1.2.3' \
-F 'autoCreate=true' \
-F bom=@bom.json \
http://localhost:8080/api/v1/bom
```

Portfolio Access Control: Assigning Access

acl

PUT	/v1/acl/mapping	Adds an ACL mapping
DELETE	/v1/acl/mapping/team/{teamUuid}/project/{projectUuid}	Removes an ACL mapping
GET	/v1/acl/team/{uuid}	Returns the projects assigned to the specified team

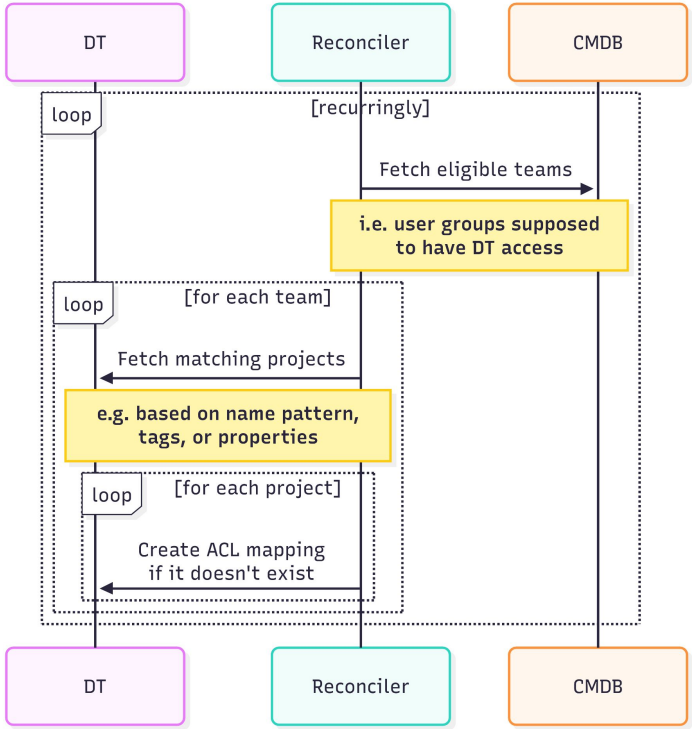
The REST API provides endpoints for assigning, revoking, and listing project access.

Portfolio Access Control: Assigning Access

Custom reconciler that ensures desired access mappings.

Well-maintained CMDB is ideal, but any mapping will do the job.

Need to pick a convention how to identify projects of a team.



WAIT, BATCH RECONCILIATION???

What is this, 2004?!

Portfolio Access Control: Assigning Access

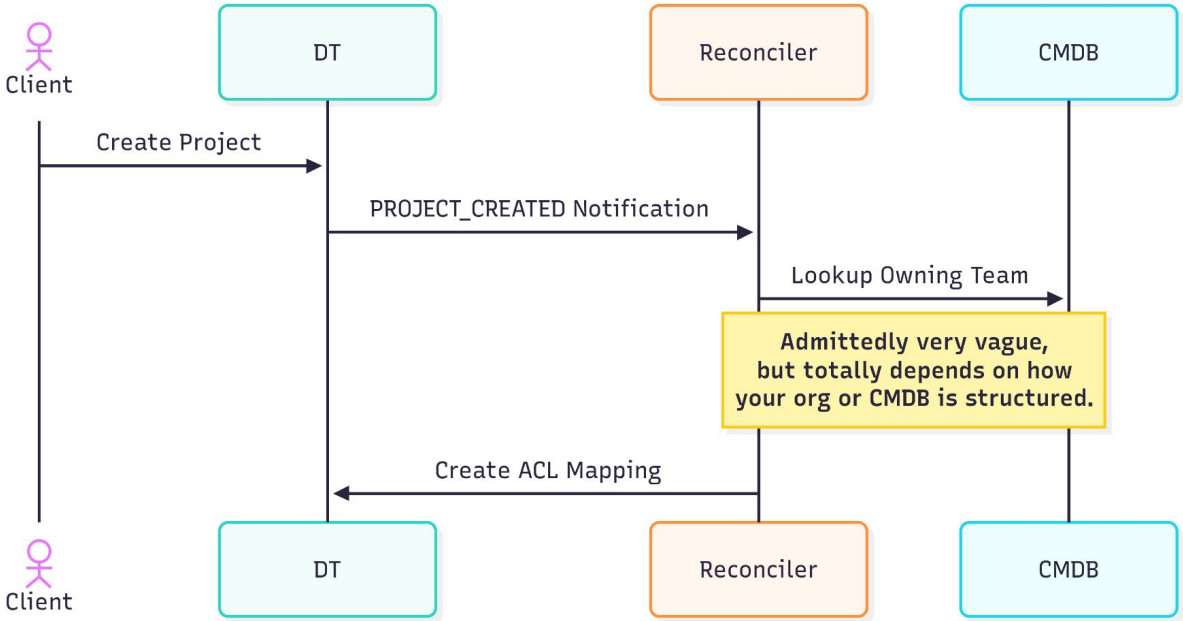
Use the community-maintained Terraform provider if you prefer an IaC approach.

```
resource "dependencytrack_project" "example" {
  name = "Example Project"
}

resource "dependencytrack_team" "example" {
  name = "Example Team"
}

resource "dependencytrack_acl_mapping" "example" {
  team    = dependencytrack_team.example.id
  project = dependencytrack_project.example.id
}
```

https://github.com/SolarFactories/terraform-provider-dependencytrack

Portfolio Access Control: Assigning Access

Leverage PROJECT_CREATED notifications to assign correct teams in neartime.



Portfolio Access Control: When to Use

- You are a large organization with silo-ed teams
- You want to let 3rd parties access a subset of projects
- You want to encourage autonomy of teams,
  but don’t want them to modify projects they don’t own
- You want to enforce principle of least privilege

Gaps & Limitations

Gaps & Limitations

- Authentication
  - Users can’t create API keys for themselves
- Authorization
  - Permissions are coarse, more granularity is desirable
- Portfolio Access Control
  - While majority of endpoints enforce it, not all do
  - Access must be granted for every version of a project
  - Project access is not inherited by child projects
  - Dashboard metrics don’t reflect limited access
  - Administration UI is finicky when dealing with many projects
  - Only admins can bypass access checks
  - Performance penalty

Planned Improvements

Planned Improvements: May Community Meeting

https://youtu.be/e-KkXkay0zA?t=834

Planned Improvements

- Ensure full enforcement of portfolio access control on all endpoints ✅
- Inheritance support for project access ✅
- Dedicated permission for portfolio access control bypass ✅
- Make it possible for project access to be assigned *for all versions*
- More fine-grained permissions (i.e., separate read from write) ✅
- Ability to assign different permission sets for different projects ⏳
- API keys for users

(Some of this may happen post v5 release)

Q&A