ONAP static code analysis by Coverity Scan

Introduction & Setup

ONAP Joint Subcommittee Meeting · Antwerp, Belgium · September 26–27, 2019

Artem Naluzhnyy <A.Naluzhnyy@samsung.com> · Samsung R&D Institute Poland

What it's about

Coverity Scan – static code analysis SaaS

- Free for Open Source projects; used by:
  - Linux kernel
  - LibreOffice
  - FreeBSD OS
- Low false positive ratio
- Shows events chain contributing to a defect
- Does not require source/build code changes to run
- Supported languages:
  - Java
  - JavaScript / TrueScript
  - Python
  - Scala
  - C / C++
  - and more…

Critical checkers

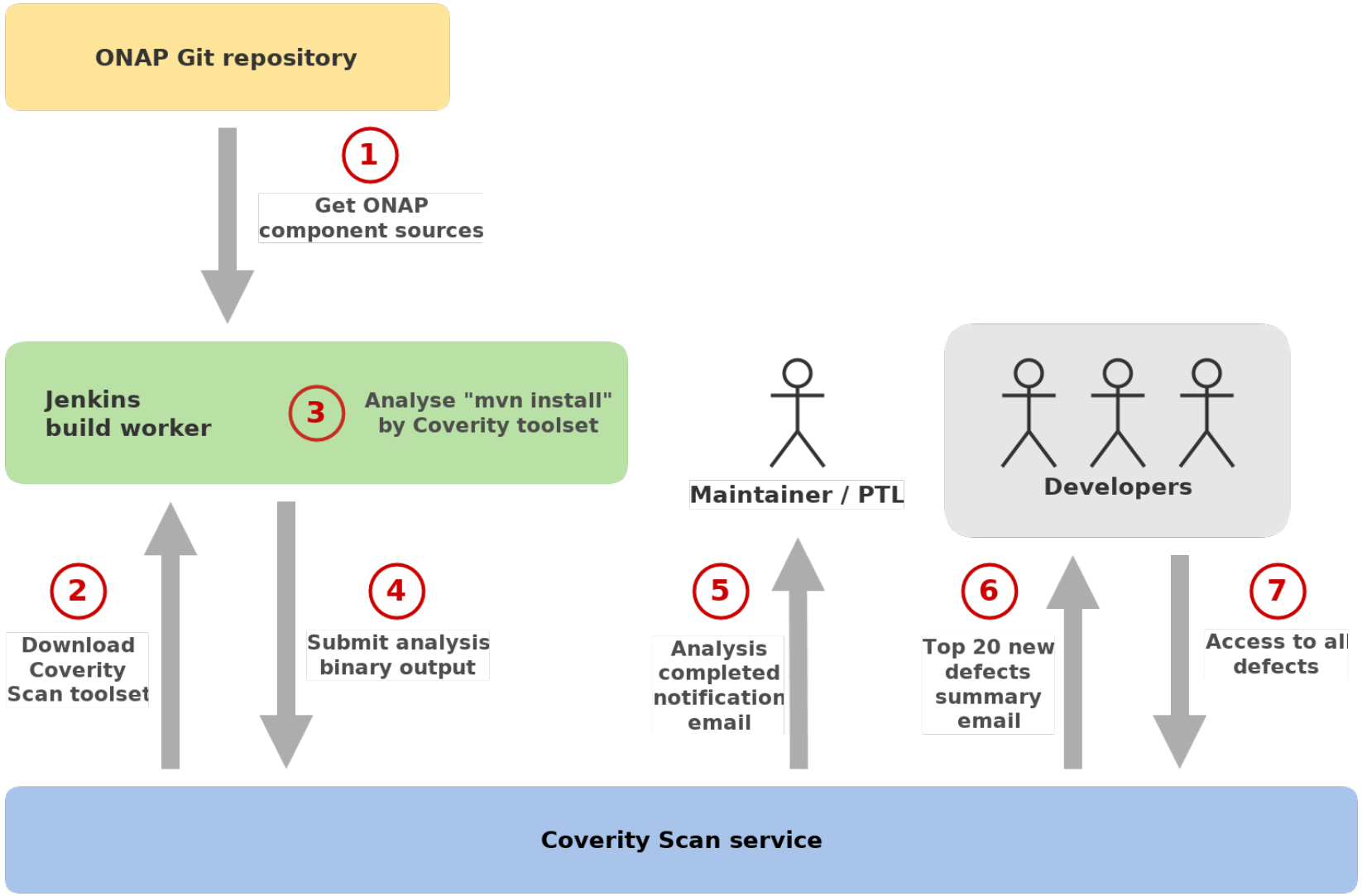
- API usage errors
- Best practice coding errors
- Buffer overflows
- Build system issues
- Class hierarchy inconsistencies
- Code maintainability issues
- Concurrent data access violations
- Control flow issues
- Cross-site request forgery (CSRF)
- Cross-site scripting (XSS)
- Deadlocks
- Error handling issues
- Hard-coded credentials
- Incorrect expression
- Insecure data handling
- Integer handling issues
- Integer overflows
- Memory corruptions
- Illegal memory accesses
- Null pointer dereferences
- Path manipulation
- Performance inefficiencies
- Program hangs
- Race conditions
- Resource leaks
- Rule violations
- Security best practices violations
- Security misconfigurations
- SQL injection
- Uninitialized members

- See also Coverity checkers [registration required]

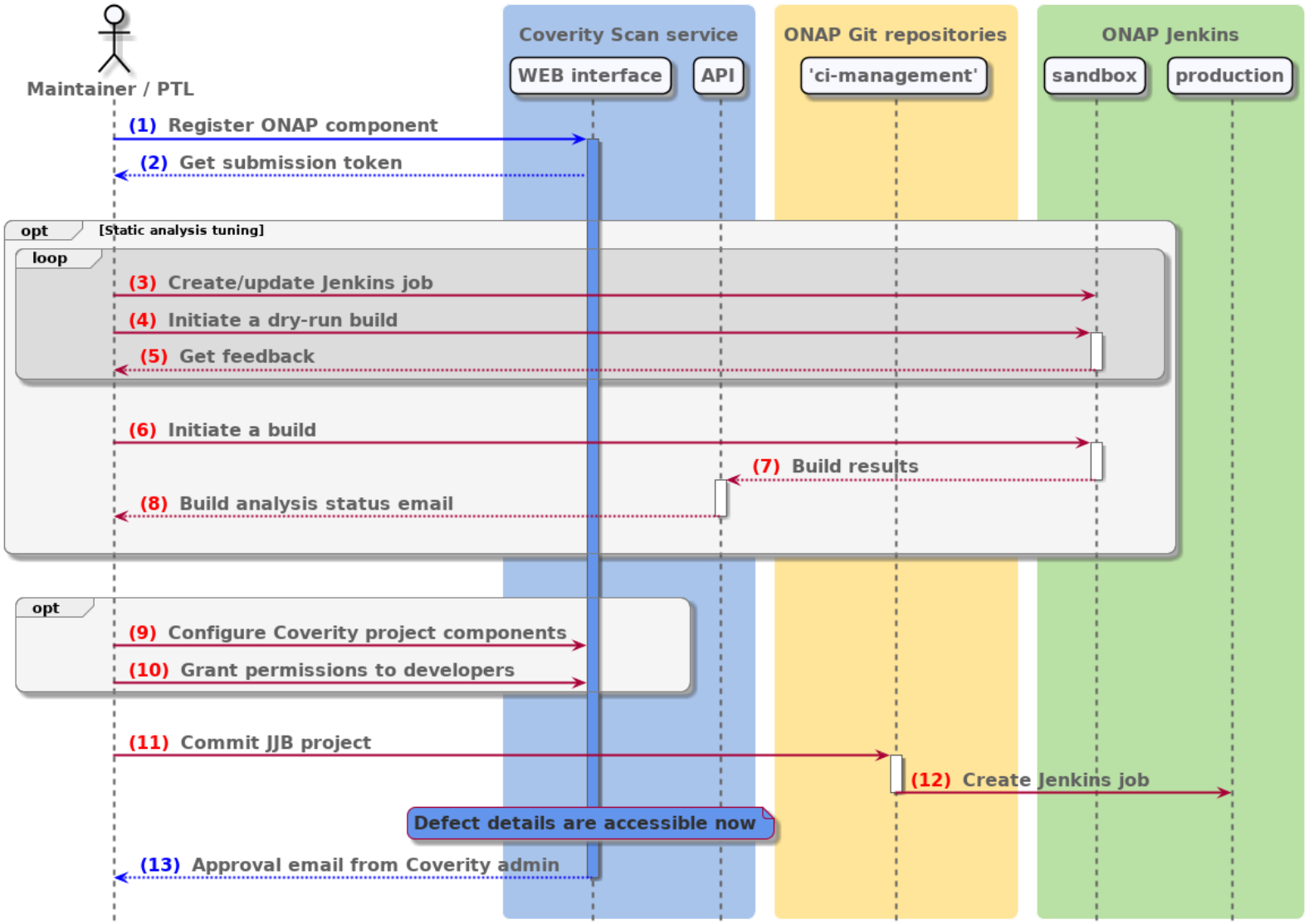
Concepts

- Coverity **project** ⇔ ONAP repository
  - Naming conventions example for "**sdc/dcae-d/fe**" ONAP repo
    → "**onap-sdc-dcae-d-fe**" Coverity project
    → "**sdc-dcae-d-fe-coverity**" ONAP Jenkins job
- Coverity **project component** ⇔ Subset of ONAP repo sources (e.g. "BE" or “FE")
  - Developers may subscribe to specific component defects only
- Coverity Scan **user roles**
  - *"Contributor/Member"* → ONAP developers (review&comment defects)
  - *"Maintainer/Owner"* → ONAP PTL / admin (configure components, grant permissions to developers)
- Coverity Scan **quotas** (1M+ LOC → 1 build per day)
- See also Coverity **glossary** [registration required]

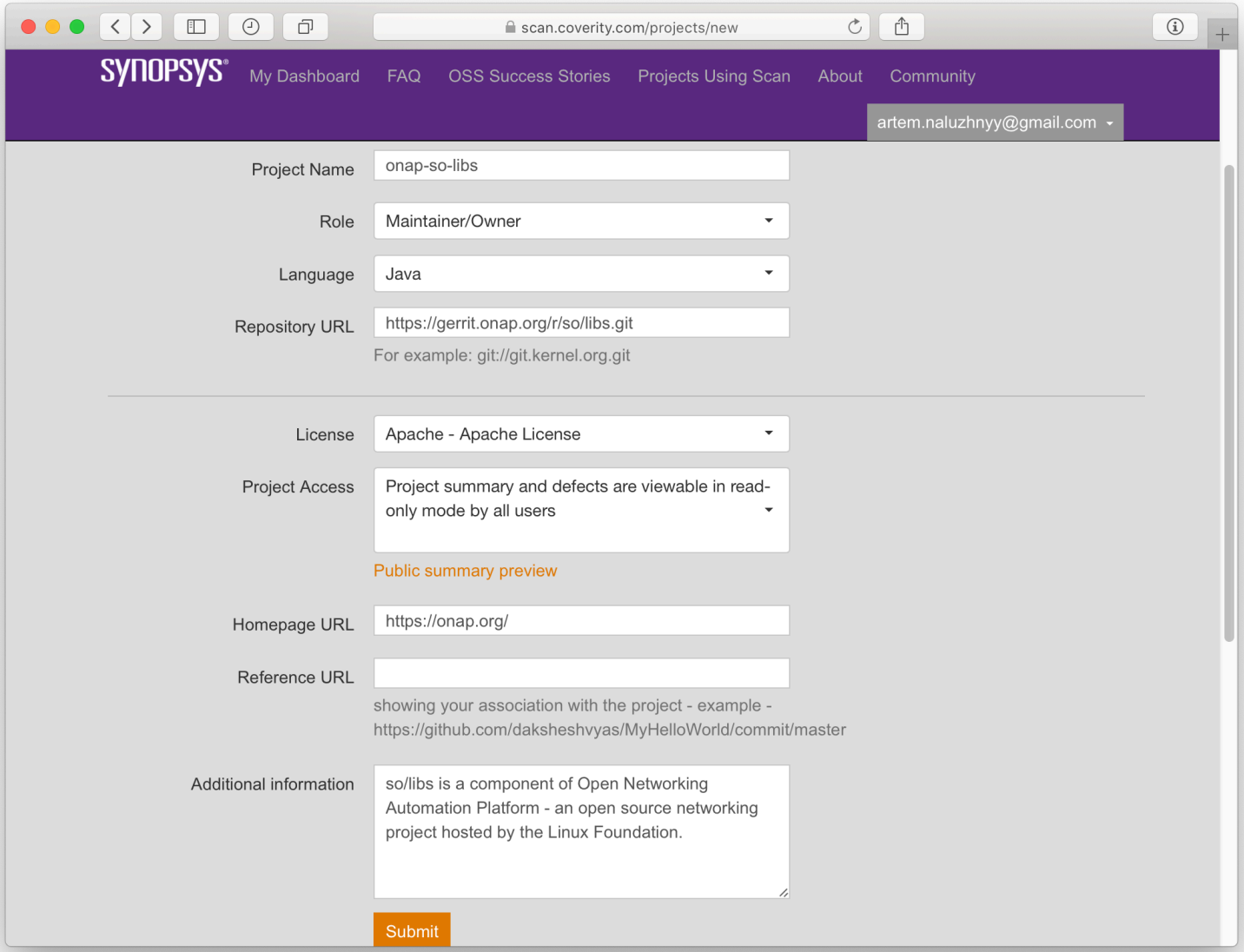
Architecture overview

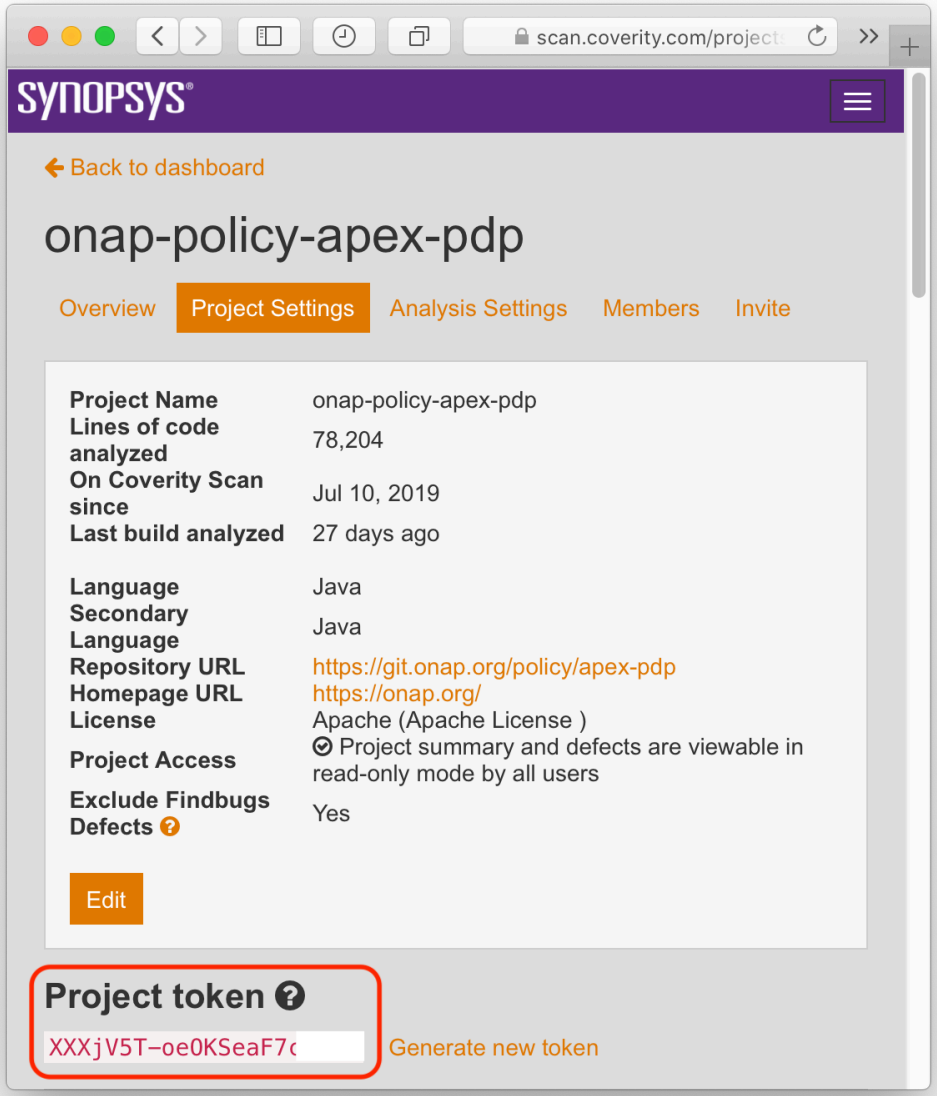


Setup workflow overview



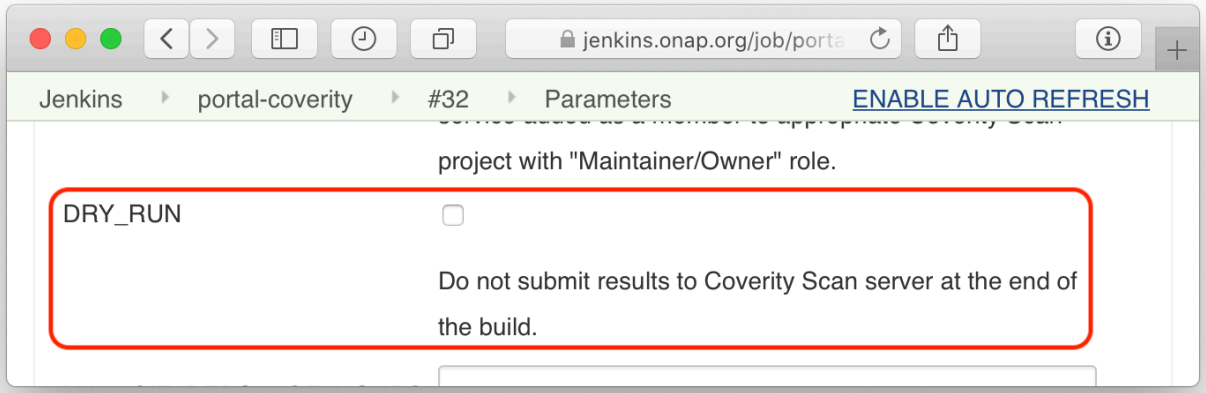
Setup workflow: Coverity project registration



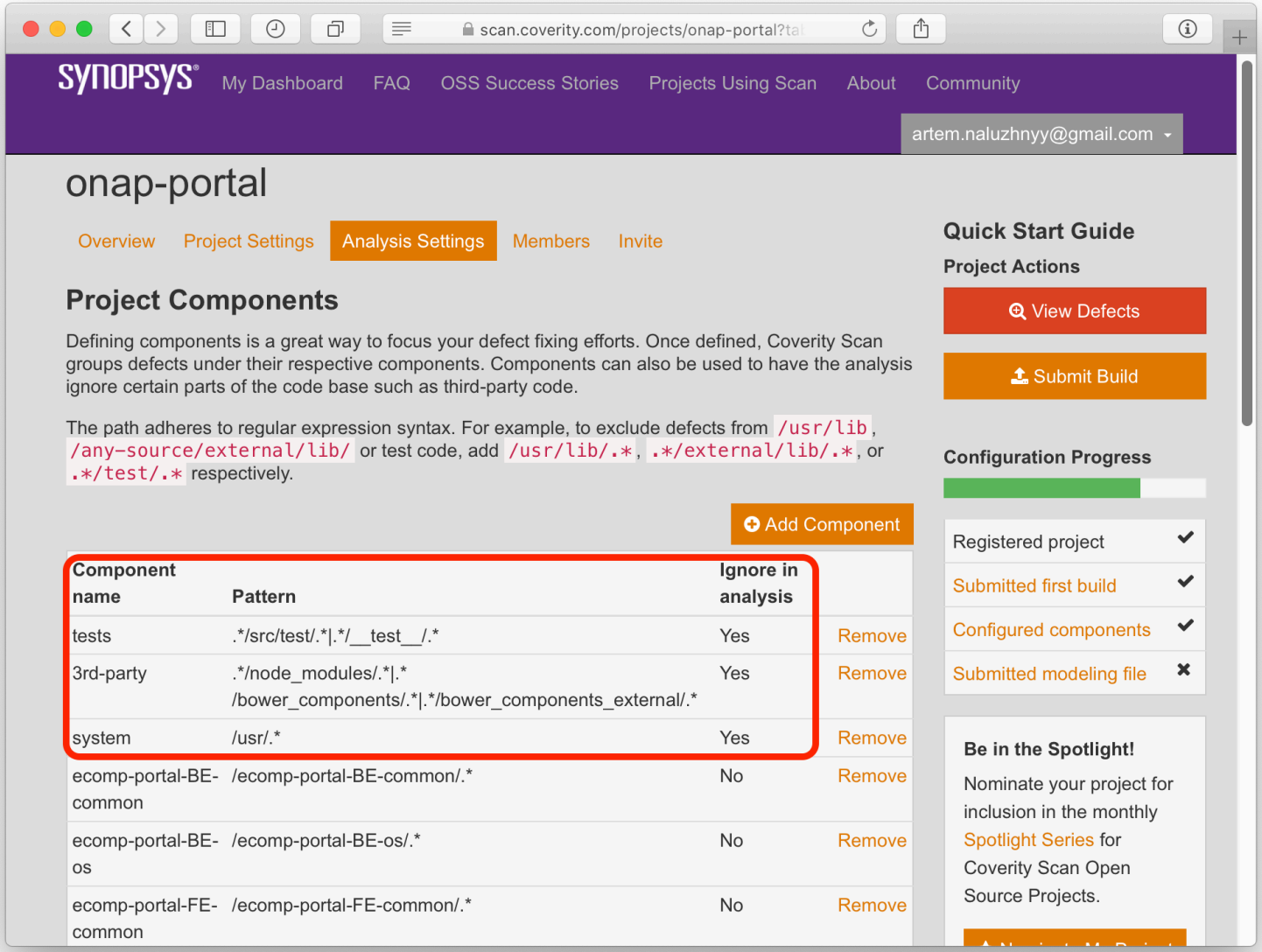


Setup workflow: ONAP Jenkins configuration

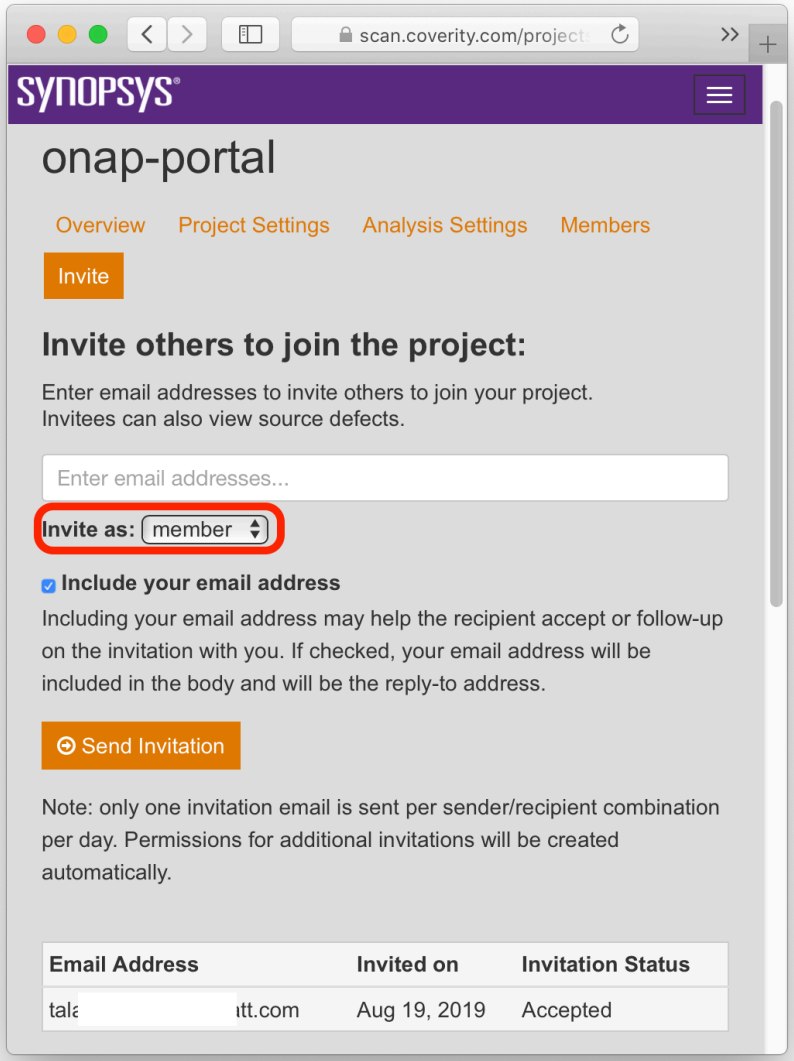
```
- project:
    name: 'portal-coverity'
    jobs:
        - 'onap-gerrit-maven-coverity'
    cron: '@daily'
    max-git-repo-age-hours: 48
    build-node: 'ubuntu1604-builder-4c-4g'
    project: 'portal'
    project-name: 'portal'
    branch: 'master'
    mvn-settings: 'portal-settings'
    mvn-params: '-Dmaven.test.skip=true'
    #mvn-params: '-DskipTests'
    coverity-project-name: 'onap-portal'
    coverity-token: 'SrGGJp9T1nn2sF72XQ'
    coverity-user-email: 'my.email@example.com'
    coverity-search-paths: >
        ecomp-portal-FE-os/client
        ecomp-portal-FE-os/mock
        ecomp-portal-FE-os/server
        ecomp-portal-widget-ms
        ecomp-portal-FE-common/client/app
    coverity-search-exclude-regexs: >
        /node_modules/
        /bower_components/
        /bower_components_external/
    dry-run: false
```



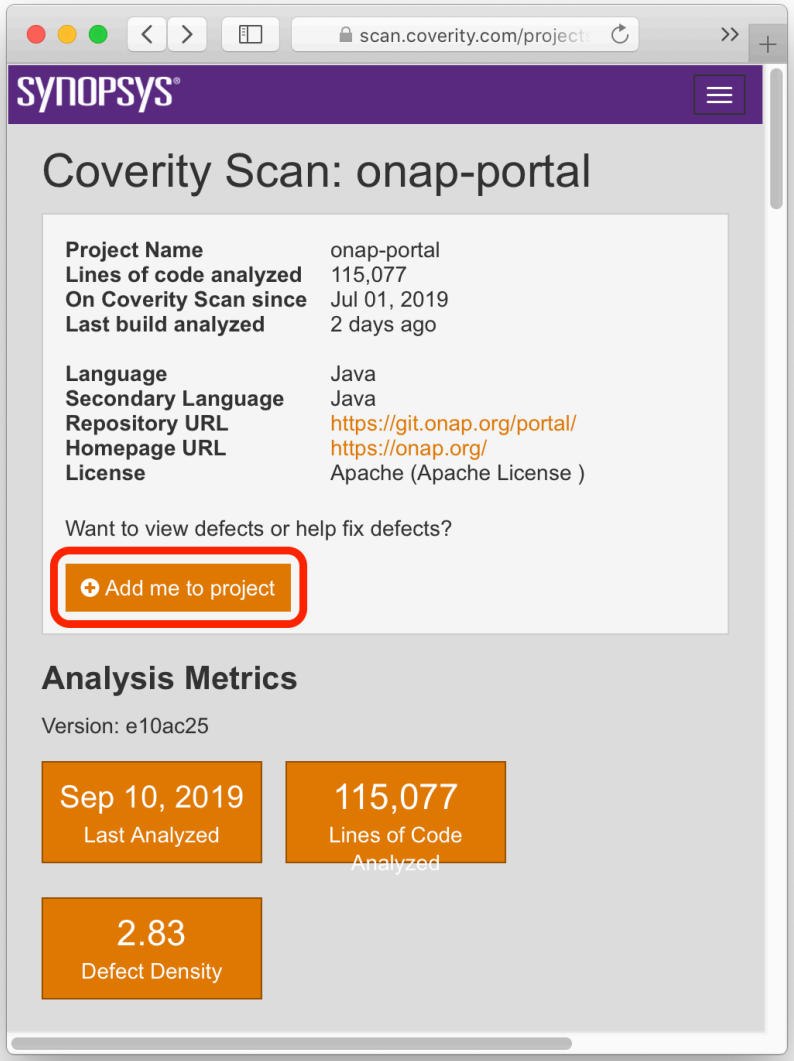
Setup workflow: Define project components



Setup workflow: Grant permissions to developers



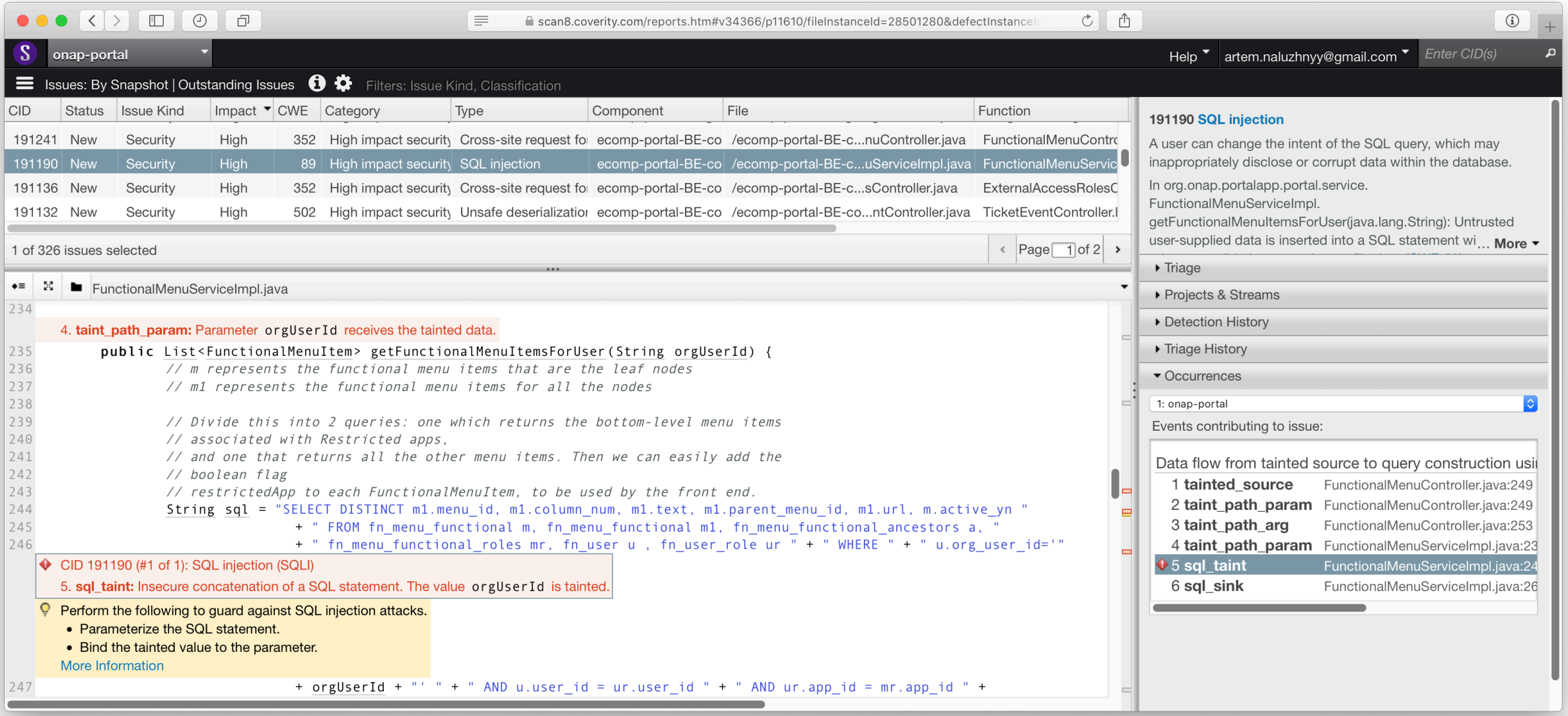
OR



Setup workflow: Troubleshooting

- Jenkins build logs:
  - **`"cov-int/coverity-scan-analysed-files.txt"`**
    → files sent for analysis
  - **`"cov-int/scm-untracked-files.txt"`**
    → 3rd-party and auto-generated sources
  - **`"cov-int/failed_jsp/*"`**
    → errors in `.jsp` files
  - **`"cov-int/build-log.txt"`**
    → Coverity toolset build log

Interface for developers



Coverity Scan service issues

- Language support:
  - *Go* is coming
  - *Kotlin* is on the roadmap
  - No support of *Clojure, Erlang* and *Lua*
- Missing source code branches support
- The service is under maintenance at the moment (2019–09–19):
  - WEB/API may be unavailable or read-only
  - specific features may be disabled

Implementation status

Done

- "**`onap-gerrit-maven-coverity`**" JJB job template [CIMAN-260]
- Analysed ONAP components
- Active Jenkins jobs
- Wiki page

To Do

- Cover more ONAP components
- Guide/demo for developers
- Optimize JJB template:
  - bandwidth & build time
  - secure Coverity tokens

Open questions

- Who should manage Coverity submission errors?
- How to manage Coverity Scan project tokens in our Jenkins?

- Should we launch tests by default?
- Should we analyse test sources by default?

- Should we analyse 3rd-party sources by default?
- Should we analyse auto-generated sources by default?

Need help?

- Check **ONAP Wiki**
- Assign Jira ticket to **@Naluzhnyy**
- Ask **Coverity Scan community**
- Contact <**scan-admin@coverity.com**>