

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...](#)

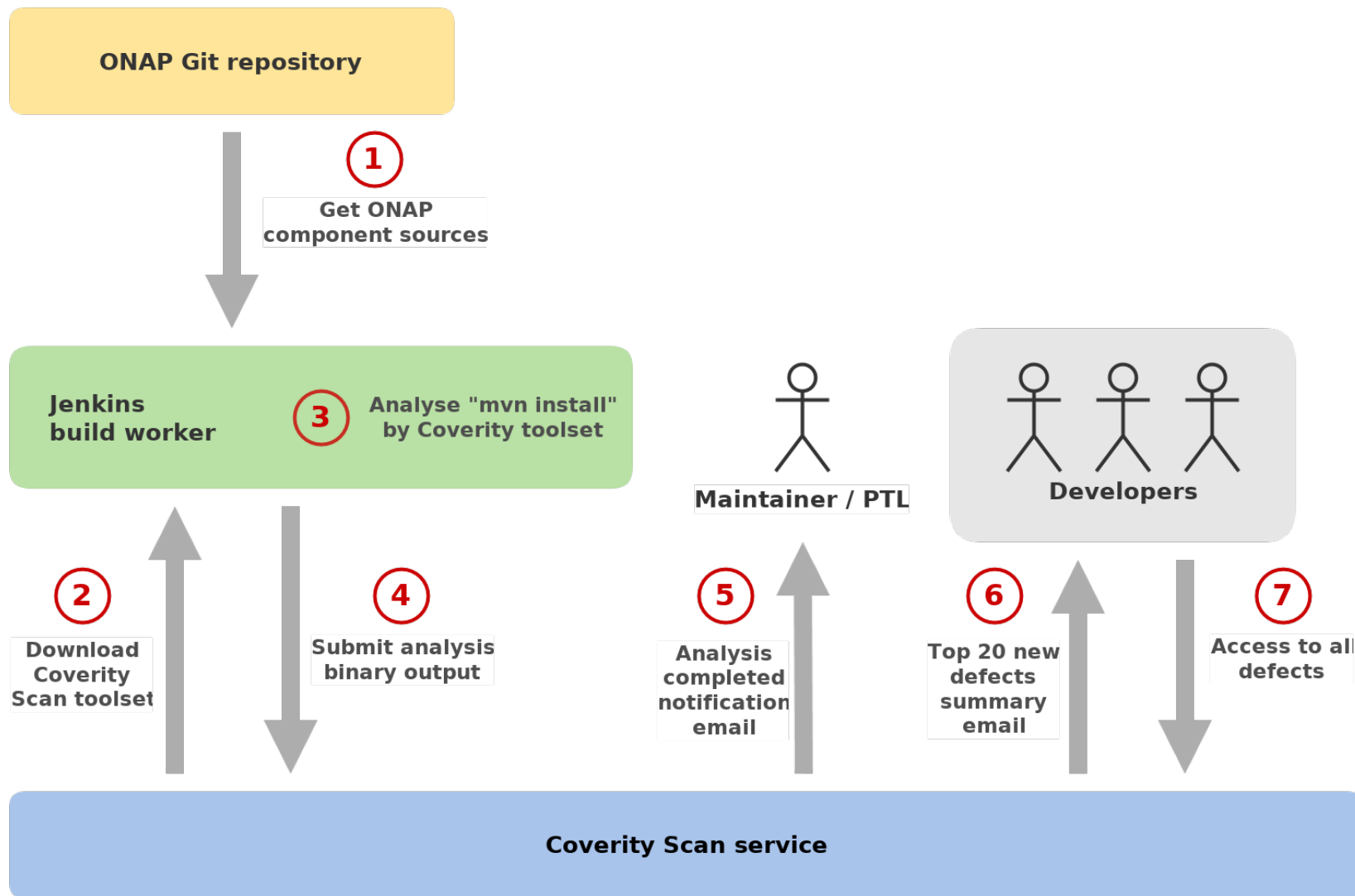
Critical checkers

- 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]

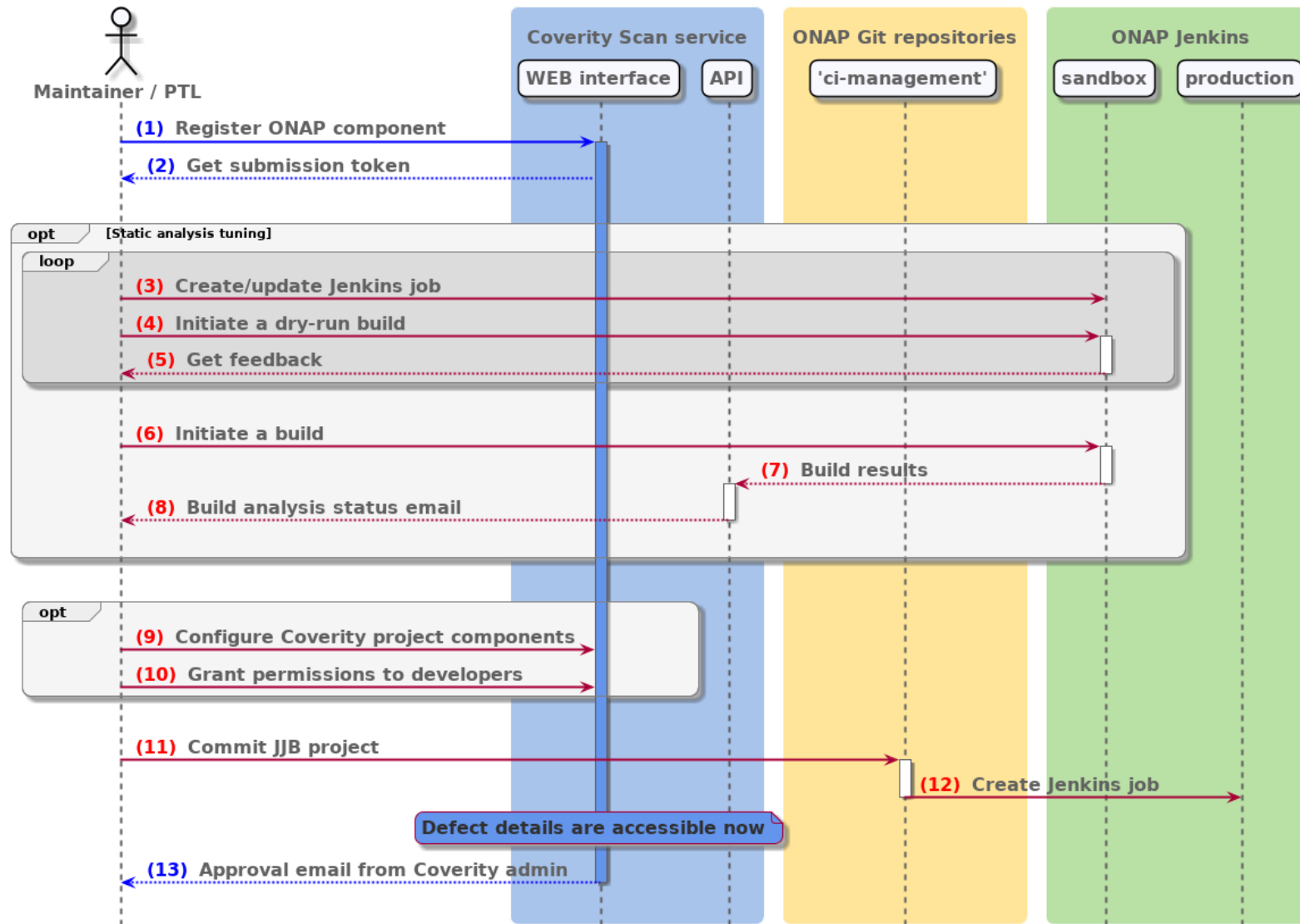
Concepts

- Coverity **project** ⇔ ONAP repository
 - Naming conventions example for "**sd**c/**dcae-d**/**fe**" ONAP repo
 - "**onap-sdc-dcae-d-fe**" Coverity project
 - "**sd**c-**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

The screenshot shows the 'New Project' registration page on the Synopsys Coverity website. The browser address bar shows 'scan.coverity.com/projects/new'. The user is logged in as 'artem.naluzhnyy@gmail.com'. The form contains the following fields:

- Project Name:** onap-so-libs
- Role:** Maintainer/Owner
- Language:** Java
- Repository URL:** https://gerrit.onap.org/r/so/libs.git
- License:** Apache - Apache License
- Project Access:** Project summary and defects are viewable in read-only mode by all users
- Homepage URL:** https://onap.org/
- Reference URL:** (empty)
- Additional information:** so/libs is a component of Open Networking Automation Platform - an open source networking project hosted by the Linux Foundation.

A 'Submit' button is located at the bottom right of the form.

The screenshot shows the 'Project Settings' page for the project 'onap-policy-apex-pdp'. The browser address bar shows 'scan.coverity.com/project'. The page includes a navigation menu with 'Overview', 'Project Settings', 'Analysis Settings', 'Members', and 'Invite'. The 'Project Settings' tab is active.

Project Details:

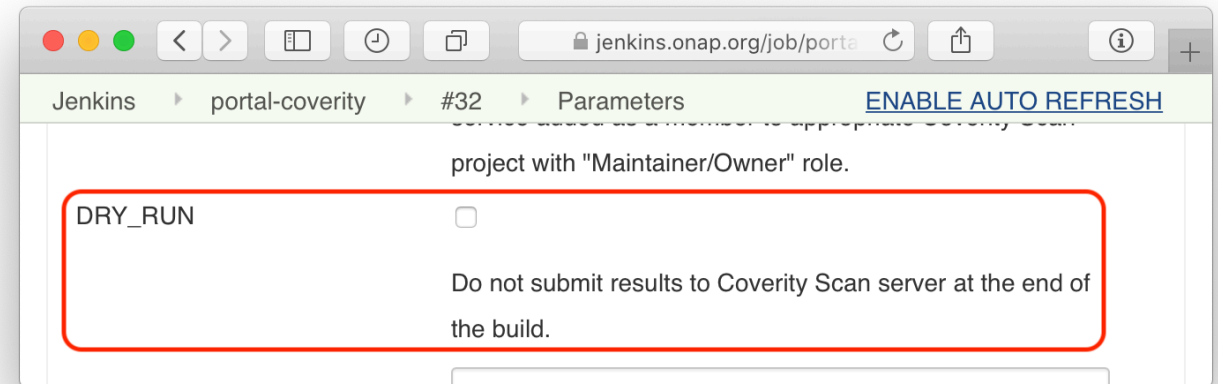
- Project Name:** onap-policy-apex-pdp
- Lines of code analyzed:** 78,204
- On Coverity Scan since:** Jul 10, 2019
- Last build analyzed:** 27 days ago
- Language:** Java
- Secondary Language:** Java
- Repository URL:** https://git.onap.org/policy/apex-pdp
- Homepage URL:** https://onap.org/
- License:** Apache (Apache License)
- Project Access:** Project summary and defects are viewable in read-only mode by all users
- Exclude Findbugs Defects:** Yes

An 'Edit' button is located below the project details.

Project token: XXXjV5T-oe0KSeaF7c (highlighted with a red box). A 'Generate new token' button is located to the right of the token.

Setup workflow: ONAP Jenkins configuration

```
1 - project:
2   name: 'portal-coverity'
3   jobs:
4     - 'onap-gerrit-maven-coverity'
5     cron: '@daily'
6     max-git-repo-age-hours: 48
7     build-node: 'ubuntu1604-builder-4c-4g'
8     project: 'portal'
9     project-name: 'portal'
10    branch: 'master'
11    mvn-settings: 'portal-settings'
12    mvn-params: '-Dmaven.test.skip=true'
13    #mvn-params: '-DskipTests'
14    coverity-project-name: 'onap-portal'
15    coverity-token: 'SrGGJp9T1nn2sF72XQ'
16    coverity-user-email: 'my.email@example.com'
17    coverity-search-paths: >
18      ecomp-portal-FE-os/client
19      ecomp-portal-FE-os/mock
20      ecomp-portal-FE-os/server
21      ecomp-portal-widget-ms
22      ecomp-portal-FE-common/client/app
23    coverity-search-exclude-regexs: >
24      /node_modules/
25      /bower_components/
26      /bower_components_external/
27    dry-run: false
```



Setup workflow: Define project components

The screenshot shows the Synopsys Coverity Scan web interface. The page title is 'onap-portal' and the user is logged in as 'artem.naluzhnyy@gmail.com'. The navigation menu includes 'My Dashboard', 'FAQ', 'OSS Success Stories', 'Projects Using Scan', 'About', and 'Community'. The main content area is titled 'Project Components' and includes a sub-menu with 'Overview', 'Project Settings', 'Analysis Settings', 'Members', and 'Invite'. A text block explains that defining components helps focus defect fixing efforts and that paths follow regular expression syntax. Below this is a table of existing components, with the first three rows highlighted by a red box. To the right, there are buttons for 'View Defects' and 'Submit Build', and a 'Configuration Progress' section showing the status of various project setup steps.

Project Components

Defining components is a great way to focus your defect fixing efforts. Once defined, Coverity Scan groups defects under their respective components. Components can also be used to have the analysis ignore certain parts of the code base such as third-party code.

The path adheres to regular expression syntax. For example, to exclude defects from `/usr/lib`, `/any-source/external/lib/` or test code, add `/usr/lib/.*`, `.*external/lib/.*`, or `.*test/.*` respectively.

[+ Add Component](#)

Component name	Pattern	Ignore in analysis	
tests	<code>./src/test/.*/_test_.*</code>	Yes	Remove
3rd-party	<code>./node_modules/.*</code> <code>/bower_components/.*bower_components_external/.*</code>	Yes	Remove
system	<code>/usr/.*</code>	Yes	Remove
ecomp-portal-BE-common	<code>/ecomp-portal-BE-common/.*</code>	No	Remove
ecomp-portal-BE-os	<code>/ecomp-portal-BE-os/.*</code>	No	Remove
ecomp-portal-FE-common	<code>/ecomp-portal-FE-common/.*</code>	No	Remove

Quick Start Guide

Project Actions

- [View Defects](#)
- [Submit Build](#)

Configuration Progress

- Registered project ✓
- Submitted first build ✓
- Configured components ✓
- Submitted modeling file ✗

Be in the Spotlight!

Nominate your project for inclusion in the monthly [Spotlight Series](#) for Coverity Scan Open Source Projects.

Setup workflow: Grant permissions to developers

SYNOPSIS®
onap-portal

Overview Project Settings Analysis Settings Members

Invite

Invite others to join the project:

Enter email addresses to invite others to join your project. Invitees can also view source defects.

Enter email addresses...

Invite as: member

Include your email address

Including your email address may help the recipient accept or follow-up on the invitation with you. If checked, your email address will be included in the body and will be the reply-to address.

Send Invitation

Note: only one invitation email is sent per sender/recipient combination per day. Permissions for additional invitations will be created automatically.

Email Address	Invited on	Invitation Status
tal@itt.com	Aug 19, 2019	Accepted

OR

SYNOPSIS®
Coverity Scan: onap-portal

Project Name	onap-portal
Lines of code analyzed	115,077
On Coverity Scan since	Jul 01, 2019
Last build analyzed	2 days ago

Language	Java
Secondary Language	Java
Repository URL	https://git.onap.org/portal/
Homepage URL	https://onap.org/
License	Apache (Apache License)

Want to view defects or help fix defects?

Add me to project

Analysis Metrics

Version: e10ac25

Sep 10, 2019 Last Analyzed	115,077 Lines of Code Analyzed
2.83 Defect Density	

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

The screenshot displays the ONAP portal interface. At the top, there's a navigation bar with the ONAP logo, user information (artem.naluzhnyy@gmail.com), and a search bar for CID(s). Below this is a header for 'Issues: By Snapshot | Outstanding Issues' with filter options for 'Issue Kind' and 'Classification'. A table lists several security issues, with CID 191190 highlighted. The main content area shows the details for CID 191190, including a description of the SQL injection vulnerability, a list of occurrences, and a data flow diagram. The code snippet shows a Java method `getFunctionalMenuItemsForUser` that constructs a SQL query using user-supplied data (`orgUserId`), which is identified as tainted. Remediation steps advise parameterizing the SQL statement.

CID	Status	Issue Kind	Impact	CWE	Category	Type	Component	File	Function
191241	New	Security	High	352	High impact security	Cross-site request fo	ecomp-portal-BE-co	/ecomp-portal-BE-c...nuController.java	FunctionalMenuContro
191190	New	Security	High	89	High impact security	SQL injection	ecomp-portal-BE-co	/ecomp-portal-BE-c...uServiceImpl.java	FunctionalMenuServic
191136	New	Security	High	352	High impact security	Cross-site request fo	ecomp-portal-BE-co	/ecomp-portal-BE-c...sController.java	ExternalAccessRolesC
191132	New	Security	High	502	High impact security	Unsafe deserializati	ecomp-portal-BE-co	/ecomp-portal-BE-co...ntController.java	TicketEventController.I

1 of 326 issues selected

Page 1 of 2

```
FunctionalMenuServiceImpl.java
234
235 4. taint_path_param: Parameter orgUserId receives the tainted data.
236 public List<FunctionalMenuItem> getFunctionalMenuItemsForUser(String orgUserId) {
237     // m represents the functional menu items that are the leaf nodes
238     // m1 represents the functional menu items for all the nodes
239
240     // Divide this into 2 queries: one which returns the bottom-level menu items
241     // associated with Restricted apps,
242     // and one that returns all the other menu items. Then we can easily add the
243     // boolean flag
244     // restrictedApp to each FunctionalMenuItem, to be used by the front end.
245     String sql = "SELECT DISTINCT m1.menu_id, m1.column_num, m1.text, m1.parent_menu_id, m1.url, m.active_yn "
246               + " FROM fn_menu_functional m, fn_menu_functional m1, fn_menu_functional_ancestors a, "
247               + " fn_menu_functional_roles mr, fn_user u , fn_user_role ur " + " WHERE " + " u.org_user_id='"
```

CID 191190 (#1 of 1): SQL injection (SQLI)
5. sql_taint: Insecure concatenation of a SQL statement. The value orgUserId is tainted.

Perform the following to guard against SQL injection attacks.

- Parameterize the SQL statement.
- Bind the tainted value to the parameter.

[More Information](#)

```
+ orgUserId + "' " + " AND u.user_id = ur.user_id " + " AND ur.app_id = mr.app_id " +
```

191190 SQL injection

A user can change the intent of the SQL query, which may inappropriately disclose or corrupt data within the database.

In org.onap.portalapp.portal.service.
FunctionalMenuServiceImpl.
getFunctionalMenuItemsForUser(java.lang.String): Untrusted user-supplied data is inserted into a SQL statement wi ... [More](#)

- Triage
- Projects & Streams
- Detection History
- Triage History
- Occurrences

1: onap-portal

Events contributing to issue:

Step	Source	Target
1 tainted_source	FunctionalMenuController.java:249	
2 taint_path_param	FunctionalMenuController.java:249	
3 taint_path_arg	FunctionalMenuController.java:253	
4 taint_path_param	FunctionalMenuServiceImpl.java:23	
5 sql_taint	FunctionalMenuServiceImpl.java:24	
6 sql_sink	FunctionalMenuServiceImpl.java:26	

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>](mailto:scan-admin@coverity.com)