CII status: NONRTRIC

Last Update: January 2021

blocked URL

Basics (12 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	NONRTRIC - Q1 2021	
Criteria	**Result / Proof point**	
Identification		
What is the human-readable name of the project?	Met	OSC Non-RealTime RIC (NONRTRIC)
What is a brief description of the project?	Met	The Non-RealTime RIC (RAN Intelligent Controller) is an Orchestration and Automation function described by the O-RAN Alliance for non-real-time intelligent management of RAN (Radio Access Network) functions. The primary goal of the NONRTRIC is to support non-real-time radio resource management, higher layer procedure optimization, policy optimization in RAN, and providing guidance, parameters, policies and AI/ML models to support the operation of near-RealTime RIC functions in the RAN to achieve higher-level non-real-time objectives. NONRTRIC functions include service and policy management, RAN analytics and model-training for the near-RealTime RICs. The non-RealTime RIC project provides concepts, architecture and reference implementations as defined and described by the O-RAN Alliance architecture.
What is the URL for the project (as a whole)?	Met	**Wiki**: *https://wiki.o-ran-sc.org/display/RICNR*
What is the URL for the version control repository (it may be the same as the project URL)?	Met	**Gerrit (Multiple)**: • nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric* • portal/nonrtric-controlpanel : *https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel* • sim/a1-interface : *https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface* • (it/dep : https://gerrit.o-ran-sc.org/r/admin/repos/it/dep *./nonrtric)*
What programming language(s) are used to implement the project?	Met	Mostly Java
What is the Common Platform Enumeration (CPE) name for the project (if it has one)?		No ID
Basic project website content		
The project website MUST succinctly describe what the software does (what problem does it solve?	Met	See: • **Wiki**: *https://wiki.o-ran-sc.org/display/RICNR* • **Docs**: *https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric*
The project website MUST provide information on how to: obtain, provide feedback (as bug reports or enhancements), and contribute to the software.	Met	• **Wiki**: https://wiki.o-ran-sc.org/pages/viewpage.action?pageId=20877888
The information on how to contribute MUST explain the contribution process (e.g., are pull requests used?) (URL required)	Met	**OSC**: https://wiki.o-ran-sc.org/display/ORAN/Tutorial%3A+Making+code+contributions+to+O-RAN+open+source+project **Gerrit (Multiple)**: • nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric* • portal/nonrtric-controlpanel : *https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel* • sim/a1-interface : *https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface* • (it/dep : https://gerrit.o-ran-sc.org/r/admin/repos/it/dep *./nonrtric)*

The information on how to contribute SHOULD include the requirements for acceptable contributions (e.g., a reference to any required coding standard). (URL required)	Partly Met	• Code style: https://wiki.o-ran-sc.org/display/RICNR/Code+Style
FLOSS license		
What license(s) is the project released under?	Met	Apache 2
The software produced by the project MUST be released as FLOSS.	Met	Apache 2
It is SUGGESTED that any required license (s) for the software produced by the project be approved by the Open Source Initiative (OSI).	Met	Apache 2
The project MUST post the license(s) of its results in a standard location in their source repository.	Met	Apache 2 - referenced in all code & repos
Documentation		
The project MUST provide basic documentation for the software produced by the project.	Met	• **Docs**: *https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric*
The project MUST provide reference documentation that describes the external interface (both input and output) of the software produced by the project.	Met	• **Docs**: https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric/en/latest/api-docs.html
Other		
The project sites (website, repository, and download URLs) MUST support HTTPS using TLS.	Met	• All support HTTPS

The project MUST have one or more mechanisms for discussion (including proposed changes and issues) that are searchable, allow messages and topics to be addressed by URL, enable new people to participate in some of the discussions, and do not require client-side installation of proprietary software.	Met	• **Meetings** (Weekly): *https://wiki.o-ran-sc.org/display/RICNR/Meetings*
		• **Email List:**
		  ◦ OSC "discuss" mailing list: *https://lists.o-ran-sc.org/g/discuss*: *discuss@lists.o-ran-sc.org*
		  ◦ Please use hashtag ***#nonrtric***
		• **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*
		• **Dicussions**: *https://wiki.o-ran-sc.org/display/RICNR/Discussions*

The project SHOULD provide documentation in English and be able to accept bug reports and comments about code in English.	Met	• All documentation in English
		• Bugs/Feedback etc accepted by:
		  ◦ **Email List:**
		  ▪ OSC "discuss" mailing list: *https://lists.o-ran-sc.org/g/discuss*: *discuss@lists.o-ran-sc.org*
		  ▪ Please use hashtag ***#nonrtric***
		  ◦ **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*
		• **OSC Guidelines**: https://wiki.o-ran-sc.org/display/ORAN/Reporting+Bugs

Change Control (9 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	NONRTRIC - Q1 2021
Criteria	**Result / Proof point**
Public version-controlled source repository	

The project MUST have a version-controlled source repository that is publicly readable and has a URL.	Met	• **Gerrit (Multiple)**: ◦ nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric* ◦ portal /nonrtric-controlpanel : *https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel* ◦ sim/a1-interface : *https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface* ◦ (it/dep : https://gerrit.o-ran-sc.org/r/admin/repos/it/dep . */nonrtric)*
The project's source repository MUST track what changes were made, who made the changes, and when the changes were made.	Met	• **Gerrit (Multiple)**: ◦ nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric* ◦ portal /nonrtric-controlpanel : *https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel* ◦ sim/a1-interface : *https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface* ◦ (it/dep : https://gerrit.o-ran-sc.org/r/admin/repos/it/dep . */nonrtric)* • **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*

To enable collaborative review, the project's source repository MUST include interim versions for review between releases; it MUST NOT include only final releases.	Met	• **Gerrit (Multiple)**:
		  ◦ nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric*
		  ◦ portal /nonrtric-controlpanel : *https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel*
		  ◦ sim/a1-interface : *https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface*
		  ◦ (it/dep : https://gerrit.o-ran-sc.org/r/admin/repos/it/dep . */nonrtric)*
		• **Binaries (Release, RC, Snapshot):**
		  ◦ Available in OSC repositories https://nexus3.o-ran-sc.org/ (LF sign-in required to browse)
		  ▪ Maven
		  ▪ Docker

It is SUGGESTED that common distributed version control software be used (e.g., git) for the project's source repository.	Met	• **Gerrit (Multiple)**:
		  ◦ nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric*
		  ◦ portal /nonrtric-controlpanel : *https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel*
		  ◦ sim/a1-interface : *https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface*
		  ◦ (it/dep : https://gerrit.o-ran-sc.org/r/admin/repos/it/dep . */nonrtric)*

Unique version numbering		

The project results MUST have a unique version identifier for each release intended to be used by users	Met	• As can be seen in OSC repositories https://nexus3.o-ran-sc.org/ (LF sign-in required to browse)
		  ◦ Maven
		  ◦ Docker

It is SUGGESTED that the Semantic Versioning (SemVer) format be used for releases.	Met	• As can be seen in OSC repositories https://nexus3.o-ran-sc.org/ (LF sign-in required to browse)
		  ◦ Maven
		  ◦ Docker
		• Also seen in Release notes:
		  ◦ https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric/en/latest/release-notes.html
		  ◦ https://docs.o-ran-sc.org/projects/o-ran-sc-sim-a1-interface/en/latest/release-notes.html
		  ◦ https://docs.o-ran-sc.org/projects/o-ran-sc-portal-nonrtric-controlpanel/en/latest/release-notes.html

It is SUGGESTED that projects identify each release within their version control system. For example, it is SUGGESTED that those using git identify each release using git tags.	Met	• As can be seen in Gerrit:
		  ◦ nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric*
		  ◦ portal /nonrtric-controlpanel : *https://gerrit.o-ran-sc.org/r/admin/repos/portal/nonrtric-controlpanel*
		  ◦ sim/a1-interface : *https://gerrit.o-ran-sc.org/r/admin/repos/sim/a1-interface*
		  ◦ (it/dep : https://gerrit.o-ran-sc.org/r/admin/repos/it/dep . */nonrtric)*

Release notes		

The project MUST provide, in each release, release notes that are a human-readable summary of major changes in that release to help users determine if they should upgrade and what the upgrade impact will be. The release notes MUST NOT be the raw output of a version control log (e.g., the "git log" command results are not release notes). Projects whose results are not intended for reuse in multiple locations (such as the software for a single website or service) AND employ continuous delivery MAY select "N/A". (URL required)	Met	• Release notes:
		  ◦ https://docs.o-ran-sc.org/projects/o-ran-sc-nonrtric/en/latest/release-notes.html
		  ◦ https://docs.o-ran-sc.org/projects/o-ran-sc-sim-a1-interface/en/latest/release-notes.html
		  ◦ https://docs.o-ran-sc.org/projects/o-ran-sc-portal-nonrtric-controlpanel/en/latest/release-notes.html

The release notes MUST identify every publicly known vulnerability with a CVE assignment or similar that is fixed in each new release, unless users typically cannot practically update the software themselves. If there are no release notes or there have been no publicly known vulnerabilities, choose "not applicable" (N/A).		N/A

Reporting (8 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	NONRTRIC - Q1 2021	
Criteria	**Result / Proof point**	
Bug-reporting process		

The project MUST provide a process for users to submit bug reports (e.g., using an issue tracker or a mailing list). (URL required)	Met	• **Wiki**: https://wiki.o-ran-sc.org/pages/viewpage.action?pageId=20877888
		  ◦ Bugs/Feedback etc accepted by:
		  ▪ **Email List:**
		  • OSC "discuss" mailing list: *https://lists.o-ran-sc.org/g/discuss*: *discuss@lists.o-ran-sc.org*
		  • Please use hashtag ***#nonrtric***
		  ▪ **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*
		• **OSC Guidelines**: https://wiki.o-ran-sc.org/display/ORAN/Reporting+Bugs

The project SHOULD use an issue tracker for tracking individual issues.	Met	• **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*

The project MUST acknowledge a majority of bug reports submitted in the last 2-12 months (inclusive); the response need not include a fix.	Met	• As can be seen on:
		  ◦ **Email List:**
		  ▪ OSC "discuss" mailing list: *https://lists.o-ran-sc.org/g/discuss*: *discuss@lists.o-ran-sc.org*
		  ▪ Please use hashtag ***#nonrtric***
		  ◦ **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*

The project SHOULD respond to a majority (>50%) of enhancement requests in the last 2-12 months (inclusive).	Met	• As can be seen on:
		  ◦ **Email List:**
		  ▪ OSC "discuss" mailing list: *https://lists.o-ran-sc.org/g/discuss*: *discuss@lists.o-ran-sc.org*
		  ▪ Please use hashtag ***#nonrtric***
		  ◦ **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*

The project MUST have a publicly available archive for reports and responses for later searching. (URL required)	Met	JIRA: https://jira.o-ran-sc.org/projects/NONRTRIC/issues
Vulnerability report process		

The project MUST publish the process for reporting vulnerabilities on the project site. (URL required)	Met	NONRTRIC: Bugs / Feedback / Vulnerabilities
		• Bugs/Feedback/Vulnerabilities can be notified by:
		  ◦ **Email List:**
		  ▪ OSC "discuss" mailing list: *https://lists.o-ran-sc.org/g/discuss*: *discuss@lists.o-ran-sc.org*
		  ▪ Please use hashtag ***#nonrtric***
		  ◦ **JIRA**: *https://jira.o-ran-sc.org/projects/NONRTRIC/issues*
		• OSC Guidelines: https://wiki.o-ran-sc.org/display/ORAN/Reporting+Bugs
		• Any private or confidential issues can be alerted to the PTL directly.

If private vulnerability reports are supported, the project MUST include how to send the information in a way that is kept private. (URL required) Examples include a private defect report submitted on the web using HTTPS (TLS) or an email encrypted using OpenPGP. If vulnerability reports are always public (so there are never private vulnerability reports), choose "not applicable" (N/A).	Met	NONRTRIC: Bugs / Feedback / Vulnerabilities • Any private or confidential issues can be alerted to the PTL directly ◦ Support for secure/encrypted reporting can be clarified directly with the PTL
The project's initial response time for any vulnerability report received in the last 6 months MUST be less than or equal to 14 days. If there have been no vulnerabilities reported in the last 6 months, choose "not applicable" (N/A).	N/A	

Quality (13 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

Criteria	**Result / Proof point**	
Working build system		
If the software produced by the project requires building for use, the project MUST provide a working build system that can automatically rebuild the software from source code.	Met	Maven
It is SUGGESTED that common tools be used for building the software.	Met	Maven
The project SHOULD be buildable using only FLOSS tools.	Met	Maven
Automated test suite		
The project MUST use at least one automated test suite that is publicly released as FLOSS (this test suite may be maintained as a separate FLOSS project).	Met	Very comprehensive test environment and tests available as a sub-directory in nonrtric repo (not a separate project) • **Gerrit** ◦ nonrtric : *https://gerrit.o-ran-sc.org/r/admin/repos/nonrtric* ./test and ./autotest
A test suite SHOULD be invocable in a standard way for that language. For example, "make check", "mvn test", or "rake test" (Ruby).	Met	• mvn test • bash scripts

It is SUGGESTED that the test suite cover most (or ideally all) the code branches, input fields, and functionality.	Met	All versions are tested in autotest suite (above)
It is SUGGESTED that the project implement continuous integration (where new or changed code is frequently integrated into a central code repository and automated tests are run on the result).	Met	Multiple tests triggered automatically and periodically as Jenkins jobs at numerous phases of CI/CD pipeline • https://jenkins.o-ran-sc.org/view/nonrtric/ • https://jenkins.o-ran-sc.org/view/portal-nonrtric-controlpanel/ • https://jenkins.o-ran-sc.org/view/sim-a1-interface/
New functionality testing		
The project MUST have a general policy (formal or not) that as major new functionality is added to the software produced by the project, tests of that functionality should be added to an automated test suite. As long as a policy is in place, even by word of mouth, that says developers should add tests to the automated test suite for major new functionality, select "Met.	Met	All new and updated functionality is tested in autotest suite (above) All contributions should include tests, as described at NONRTRIC: Code Style
The project MUST have evidence that the test_policy for adding tests has been adhered to in the most recent major changes to the software produced by the project. Major functionality would typically be mentioned in the release notes. Perfection is not required, merely evidence that tests are typically being added in practice to the automated test suite when new major functionality is added to the software produced by the project.	Met	All new and updated functionality is tested in autotest suite (above). New test functionality can be seen in JIRA and Gerrit. Unit test coverage can be verified using Sonar • https://jenkins.o-ran-sc.org/view/nonrtric/job/nonrtric-sonar/ • https://jenkins.o-ran-sc.org/view/portal-nonrtric-controlpanel/job/portal-nonrtric-controlpanel-sonar/ • https://jenkins.o-ran-sc.org/view/sim-a1-interface/job/sim-a1-interface-tox-sonarqube/
It is SUGGESTED that this policy on adding tests (see test_policy) be *documented* in the instructions for change proposals. However, even an informal rule is acceptable as long as the tests are being added in practice.	Met	Only informal rule exists, tests are continuously added in practice All contributions should include tests, as described at NONRTRIC: Code Style
Warning flags		

The project MUST enable one or more compiler warning flags, a "safe" language mode, or use a separate "linter" tool to look for code quality errors or common simple mistakes, if there is at least one FLOSS tool that can implement this criterion in the selected language.	Met	• Sonar is used in development environment and automatically triggered by Jenkins during CI/CD process
		  ◦ Jenkins:
		  ▪ https://jenkins.o-ran-sc.org/view/nonrtric/job/nonrtric-sonar/
		  ▪ https://jenkins.o-ran-sc.org/view/portal-nonrtric-controlpanel/job/portal-nonrtric-controlpanel-sonar/
		  ▪ https://jenkins.o-ran-sc.org/view/sim-a1-interface/job/sim-a1-interface-tox-sonarqube/
		• CheckStyle & Findbugs are used in development environments

The project MUST address warnings.	Met	Sonar reports are acted upon continuously.
It is SUGGESTED that projects be maximally strict with warnings in the software produced by the project, where practical. Some warnings cannot be effectively enabled on some projects. What is needed is evidence that the project is striving to enable warning flags where it can, so that errors are detected early.	Met	All Checkstyle, Findbugs, test failures, notified issues/bugs and Sonar warnings are acted on promptly. Such issues are tracked using Jira and Gerrit (See above).

Security (16 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	NONRTRIC - Q1 2021
Criteria	**Result / Proof point**
Secure development knowledge	

The project MUST have at least one primary developer who knows how to design secure software. (See 'details' for the exact requirements.)	Met	All team members are trained to develop secure software
At least one of the project's primary developers MUST know of common kinds of errors that lead to vulnerabilities in this kind of software, as well as at least one method to counter or mitigate each of them.	Met	All team members are trained to develop secure software
Use basic good cryptographic practices		
The software produced by the project MUST use, by default, only cryptographic protocols and algorithms that are publicly published and reviewed by experts (if cryptographic protocols and algorithms are used).These cryptographic criteria do not always apply because some software has no need to directly use cryptographic capabilities.	Met	All external operational interfaces support (optional) cryptographic authentication & encryption using shared key cryptography (TLS /HTTPs)
If the software produced by the project is an application or library, and its primary purpose is not to implement cryptography, then it SHOULD only call on software specifically designed to implement cryptographic functions; it SHOULD NOT re-implement its own.	Met	Commonly used existing opensource cryptographic libraries are used
All functionality in the software produced by the project that depends on cryptography MUST be implementable using FLOSS. See the Open Standards Requirement for Software by the Open Source Initiative.	Met	Commonly used existing opensource cryptographic libraries are used
The security mechanisms within the software produced by the project MUST use default keylengths that at least meet the NIST minimum requirements through the year 2030 (as stated in 2012). It MUST be possible to configure the software so that smaller keylengths are completely disabled.These minimum bitlengths are: symmetric key 112, factoring modulus 2048, discrete logarithm key 224, discrete logarithmic group 2048, elliptic curve 224, and hash 224 (password hashing is not covered by this bitlength, more information on password hashing can be found in the crypto_password_storage criterion). See https://www.keylength.com for a comparison of keylength recommendations from various organizations. The software MAY allow smaller keylengths in some configurations (ideally it would not, since this allows downgrade attacks, but shorter keylengths are sometimes necessary for interoperability).	Met	No keys intended for production use are included. Keys needed for TLS needs to be provided at SW installation. There is no inbuilt limitations on key length etc.
The default security mechanisms within the software produced by the project MUST NOT depend on broken cryptographic algorithms (e.g., MD4, MD5, single DES, RC4, Dual_EC_DRBG), or use cipher modes that are inappropriate to the context, unless they are necessary to implement an interoperable protocol (where the protocol implemented is the most recent version of that standard broadly supported by the network ecosystem, that ecosystem requires the use of such an algorithm or mode, and that ecosystem does not offer any more secure alternative). The documentation MUST describe any relevant security risks and any known mitigations if these broken algorithms or modes are necessary for an interoperable protocol.	Met	Sample keys are RSA keys, and stored in a java keystore file (PKCS12), which is encrypted by a password. It is the responsibility of the production user to ensure their keys are compliant.
The default security mechanisms within the software produced by the project SHOULD NOT depend on cryptographic algorithms or modes with known serious weaknesses (e.g., the SHA-1 cryptographic hash algorithm or the CBC mode in SSH).	Met	Does not depend on any any particular cryptographic algorithm. A cert may be signed using SHA-1, but it is up to the cert issuer. It is the responsibility of the production user to ensure their keys are compliant. Sample certs are signed.
The security mechanisms within the software produced by the project SHOULD implement perfect forward secrecy for key agreement protocols so a session key derived from a set of long-term keys cannot be compromised if one of the long-term keys is compromised in the future.	Met	Standard TLS /HTTPS used.
If the software produced by the project causes the storing of passwords for authentication of external users, the passwords MUST be stored as iterated hashes with a per-user salt by using a key stretching (iterated) algorithm (e.g., Argon2id, Bcrypt, Scrypt, or PBKDF2). See also OWASP Password Storage Cheat Sheet).	Met	No users' passwords are handled.

The security mechanisms within the software produced by the project MUST generate all cryptographic keys and nonces using a cryptographically secure random number generator, and MUST NOT do so using generators that are cryptographically insecure.	Met	Standard/best-practice TLS /HTTPS implementations are used. Java, Jetty, Netty, Springboot.
Secured delivery against man-in-the-middle (MITM) attacks		
The project MUST use a delivery mechanism that counters MITM attacks. Using https or ssh+scp is acceptable.	Met	HTTPS/TLS used (optionally) throughout for all external operational interfaces
A cryptographic hash (e.g., a sha1sum) MUST NOT be retrieved over http and used without checking for a cryptographic signature.	Met	Standard/best-practice TLS /HTTPS implementations are used. Java, Jetty, Netty, Springboot.
Publicly known vulnerabilities fixed		
There MUST be no unpatched vulnerabilities of medium or higher severity that have been publicly known for more than 60 days.	Met (TBC)	LF is currently enabling scans across all projects. All high/med vulnerabilities, once identified, are fixed with highest priority (<60 days)
Projects SHOULD fix all critical vulnerabilities rapidly after they are reported.	Met	All high/med vulnerabilities, once identified, are fixed with highest priority (<60 days)
Publicly known vulnerabilities fixed		
is intended to limit public access. A project MAY leak "sample" credentials for testing and unimportant databases, as long as they are not intended to limit public access.	Met	All sample/provided credentials are for test/demo purposes only.

Analysis (8 Points)

(Result/Proof point (column A: enter Met/Unmet; Column B: enter relevant URLs/comments)

	NONRTRIC - Q1 2021
Criteria	**Result / Proof point**
Static code analysis	

At least one static code analysis tool (beyond compiler warnings and "safe" language modes) MUST be applied to any proposed major production release of the software before its release, if there is at least one FLOSS tool that implements this criterion in the selected language.	Met	• Sonar is used in development environment and automatically triggered by Jenkins during CI/CD process
		  ◦ Jenkins:
		  ▪ https://jenkins.o-ran-sc.org/view/nonrtric/job/nonrtric-sonar/
		  ▪ https://jenkins.o-ran-sc.org/view/portal-nonrtric-controlpanel/job/portal-nonrtric-controlpanel-sonar/
		  ▪ https://jenkins.o-ran-sc.org/view/sim-a1-interface/job/sim-a1-interface-tox-sonarqube/
		• Findbugs is used in development environment
		• NexusIQ (from Sonatype) is also used - focuses on license scan and performs a CVE scan based on versions used.
		• A per-release license scan based is performed for LF Legal - uses a LF-internal scanning.

It is SUGGESTED that at least one of the static analysis tools used for the static_analysis criterion include rules or approaches to look for common vulnerabilities in the analyzed language or environment.	Met	• Sonar is used in development environment and automatically triggered by Jenkins during CI/CD process
		  ◦ Jenkins:
		  ▪ https://jenkins.o-ran-sc.org/view/nonrtric/job/nonrtric-sonar/
		  ▪ https://jenkins.o-ran-sc.org/view/portal-nonrtric-controlpanel/job/portal-nonrtric-controlpanel-sonar/
		  ▪ https://jenkins.o-ran-sc.org/view/sim-a1-interface/job/sim-a1-interface-tox-sonarqube/
		• Findbugs and Checkstyle is used in development environments
		• NexusIQ (from Sonatype) is also used - focuses on license scan and performs a CVE scan based on versions used.
		• A per-release license scan based is performed for LF Legal - uses a LF-internal scanning.

All medium and higher severity exploitable vulnerabilities discovered with static code analysis MUST be fixed in a timely way after they are confirmed.	Met	All reports are acted upon continuously.
It is SUGGESTED that static source code analysis occur on every commit or at least daily.	Met (TBC)	Sonar is used in development environment and automatically triggered by Jenkins during CI/CD process
Dynamic code analysis		

It is SUGGESTED that at least one dynamic analysis tool be applied to any proposed major production release of the software before its release.	TBC	
It is SUGGESTED that if the software produced by the project includes software written using a memory-unsafe language (e.g., C or C++), then at least one dynamic tool (e.g., a fuzzer or web application scanner) be routinely used in combination with a mechanism to detect memory safety problems such as buffer overwrites. If the project does not produce software written in a memory-unsafe language, choose "not applicable" (N/A).	N/A	Java, Python is used.
It is SUGGESTED that the software produced by the project include many run-time assertions that are checked during dynamic analysis.	Met	Test-level assertions are checked in tests. All runtime failing assertions /exceptions are caught and logged. Any error/exception appearing in test logs or reported by users are assessed in timely manner.
All medium and higher severity exploitable vulnerabilities discovered with dynamic code analysis MUST be fixed in a timely way after they are confirmed.	Met	No exploitable vulnerabilities to our knowledge.