LF NETWORKING

WHITEPAPER

Securing Open Source 5G from End-to-End

An LF Networking publication

Table of Contents

Preface

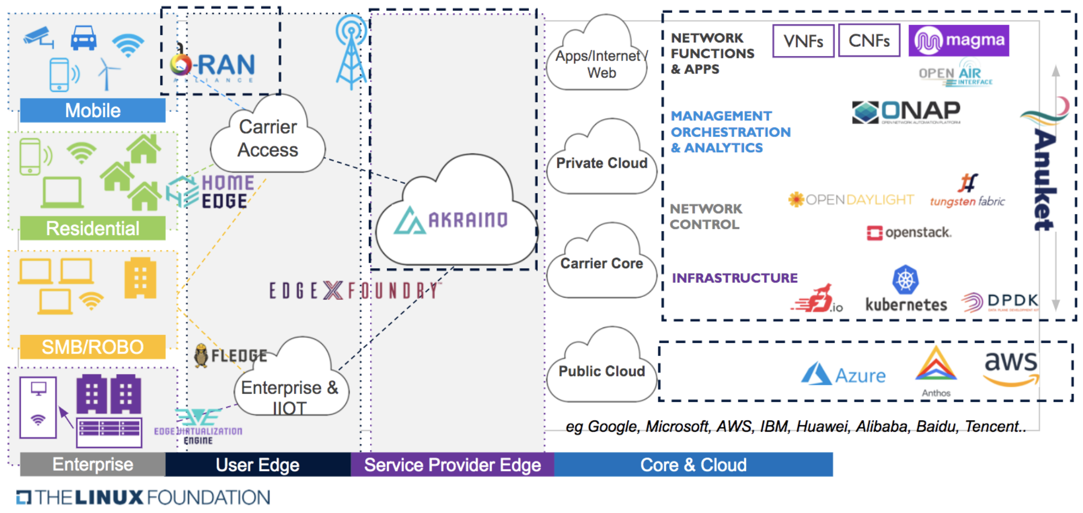
Introduction to Linux Foundation Networking (LF Networking)

The modern organization has data and users located everywhere. This is impacting the performance and security needs of IT networks across the globe, along with compelling user experiences and new revenue paths—all central to 'digital transformation'. Networking technology plays a pivotal role in this digital transformation and among them, three networking transformations will disrupt our planet from 2020-2030: 5G, edge computing, and cloud native application development. Any one of these disruptions can be overwhelming to comprehend, let alone to leverage in totality in a rapid and affordable way. Inherently, software complexity is a key challenge to implementing the full capabilities of these technologies and this is where open source comes into play. Leverage you must: adapt or perish.

LF Networking's guiding principle is that open source technology is the only viable path to truly scale software so that businesses, government agencies, education institutions, service providers—and the OEMs, ISVs and system integrators that support them—can achieve operational and revenue value in a timely and cost-effective manner. Open source software also motivates leading edge development with vulnerability detection and code design best practices with security built in from the ground up.

LF Networking is the center of gravity for collaboration—through a broad industry coalition—so the entire world can access networking innovations and achieve digital transformation. No single company could accomplish what is delivered through LF Networking. Companies of all sizes and types can rely on a breadth of commercially-ready ecosystem offerings that are based on LF Networking innovations.

Open Source Projects Being Consumed and Deployed



Key Community Benefits Include:

- **Faster time to market — with Open Interoperability.**
  Open source has now become the de facto way to build software to speed up software development. Within the LF Networking community, developers have access to the broadest set of open source networking capabilities and integration points, with projects spanning Software Defined Networking (SDN), Network Functions Virtualization (NFV), Management and Orchestration (MANO), automation, analytics, data plane acceleration, and more. They can leverage this community-driven work to develop and onboard networking software in simpler, more efficient ways. This streamlines development and speeds up time to market for network services and infrastructure, for both private and public networks alike. In turn, businesses, service providers and government agencies have the confidence that they can rely on community-built, interoperable solutions to support their end user needs.

- **Making & Saving Costs — CAPEX & OPEX.**
  Software defined networks, virtualization, disaggregation, and the dawn of brings a plethora of new opportunities for new service development and new revenue streams. This is particularly true of open source, where new innovations in open source projects are driving new frameworks and paradigms for what's possible. At the same time, operations need to run smoothly and can't afford costly or time-consuming software upgrades. Working in the LF Networking community prevents unnecessary duplication of common networking components and technology that companies need to develop and deploy, helping reduce or optimize capital and operating expenditures. By leveraging work within the community, gain simpler, more cost-efficient ways to develop and onboard software and can leverage a broad choice of vendors to reduce lock in. LF Networking's broad community and years of experience in driving open compliance efforts also offers assurance of interoperability and removes the guesswork, saving both time and money. The community is constantly improving both the requirements and the testing against these requirements. Whether leveraging a single community project building block or a more complete solution stack.

- **Speed of Innovation — Stay Cutting Edge.**
  Innovation happens faster in an open, collaborative forum where ideas are crowdsourced, shared and quickly iterated upon. The LF Networking community is fostering a new wave of open source practitioners across the entire value chain and is an innovation competency that the industry can rely on. The community fosters and incubates networking innovations that have been implemented in communications networks globally and can be used to enable new revenue-generating services and operational efficiencies across a range of industries, from retail and healthcare to manufacturing and more. Open source software, by definition, also motivates leading edge development with vulnerability detection & code design best practices. In addition, community projects are deeply integrated with the edge computing landscape and collaborate with other communities such as LF Edge to add networking capabilities and features to edge blueprints and use cases. With this open access to incredible networking innovations, retailers can find new ways to optimize inventory, manufacturers can improve defect detection and connect their factories for real-time decision making, and many other sectors can benefit.

LF Networking Security

Open source software motivates leading edge development with vulnerability detection and code design best practices with security built in from the ground up. LF Networking views open source security as a continuous improvement process that tracks to a project's maturity level and phase of adoption. Throughout the project lifecycle, additional investments are made in security process definition, security audits, and security threat modeling. LFN projects aim to develop security best practices in areas such as Requirements & Architecture, Security Framework, Vulnerability Testing, and Secure Configuration. License scans are run against several LFN project code bases monthly with results and SBOMS published in public repos.

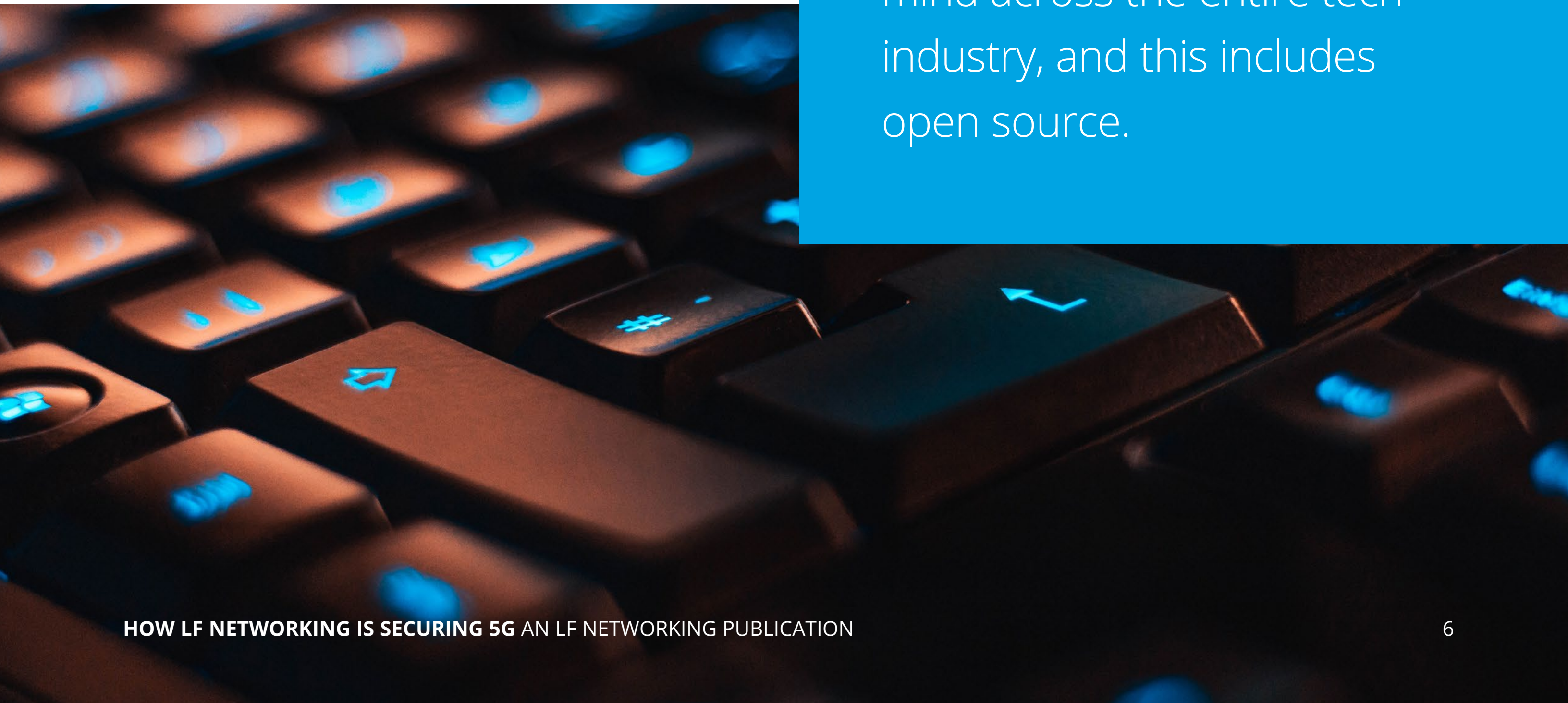
Introduction

Cybersecurity is top of mind across the entire tech industry, and this includes open source. With such tremendous growth in the past year alone (that includes 73% growth of component downloads; 6M new versions added in a year; 37M open source software component versions available) it's no surprise there has been a 650% year-over-year increase in cyberattacks aimed at open source software suppliers.

The problem has caught the attention of the United States White House, with two recent Executive Orders: an **Executive Order on Improving the Nation's Cybersecurity** issued in May of 2021, and in February 2022, an **EO on supply chains**. In January of 2022, The White House convened an important cross-section of the Open Source developer and commercial ecosystem along with leaders and experts of many U.S. federal agencies to identify the challenges present in the open source software supply chain and share ideas on ways to mitigate risk and enhance resilience.

At the meeting, the **Linux Foundation** and the Open Source Security Foundation (OpenSSF) represented their hundreds of communities and projects by highlighting collective cybersecurity efforts and sharing their intent to work with the administration across public and private sectors. (More details on the outcomes of that meeting **can be found here**).

Cybersecurity is top of mind across the entire tech industry, and this includes open source.



Designing a Secure Architecture and Creating Best Practices

The security advantages of open source software have been widely acknowledged for years. Having more eyeballs on the code base makes it difficult to hide malicious backdoors, and easy to spot vulnerabilities. However, making the code base open is just the first step. Reality taught us that without a holistic approach to security, that encompases the entire software lifecycle, vulnerabilities can go unnoticed for years. One recent example is the exploitable weaknesses of the popular log4j open source library. The LFN is committed to implementing such an end-to-end approach for software security, with security measures that begin with the design phase of the software and go through the implementation and testing phases. LFN communities are working diligently to provide our end users peace of mind, knowing the software was designed and tested to provide a secure network infrastructure.

The security playbooks developed by the LFN were forged by fire, based on the experience of our projects in dealing with security threats. As such, they are applicable not only to open source networking, but for most open source projects, within the realms of the LF and beyond.

LF Networking Security Committees and Task Forces

One of the unique advantages of the open source software development methodology is having more eyeballs on the code base. As a result, we have a large group of industry experts who have access and can propose improvements and enhancements. Our experience indicates that having discussions in broad forums brings out the best and brightest ideas. The challenge is always keeping these discussions focused and results oriented. Several committees and task forces under the LFN were proven to be the best platforms to have such discussions and create the required best practices and procedures.

Members of the LFN project communities came together in the second half of 2021 to create the LFN Security Forum workgroup. The purpose of this forum is for cross-LFN community members to discuss anything related to security, including threat analysis, industry trends, best practices, tools, and more.

To date, the LFN Security Forum has conducted several meetings; while the initial focus is knowledge-sharing, meetings to date have included an SBOM discussion and DDos Mitigation discussion. Ad-hoc meetings are organized on a per-need basis, and the forum is actively developing best practices in between meetings, using an active mailing list and a wiki collaboration space.

ONAP, one of LFN's most mature projects, created a security response strategy in 2018, which includes two parts: 1) The management of identified vulnerabilities, which is handled by the vulnerability management response team; and 2) the coordination and identification of necessary security related activities, handled by the security sub-committee (SECCOM). Details on the ONAP vulnerability management process are available here.

The ONAP security sub-committee (SECCOM) identifies and creates proposals related to security in ONAP. As one example, it has created the proposal for the Vulnerability management procedures which are now in effect. The ongoing efforts of the ONAP security sub-committee explore more proactive security activities such as the adoption of the OpenSFF badging process. The SECCOM meets weekly and maintains a wiki with recommendations on security, consisting of a few parts:

ONAP Security Recommendation Development where recommendations are developed; once the group feels they are ready,

- Adding that recommendation to the ONAP Security Best Practices list.
- Specific corrective action recommendations are made to the ONAP Technical Steering Committee (TSC) which when approved are mandated in every subsequent release.

All projects participating in an ONAP release must meet the security best practices approved by the TSC. Chief among these is the requirement that all vulnerable open source packages used by ONAP be upgraded in each release. If for any reason a new security requirement cannot be met by one of the ONAP sub-projects for a particular release, a waiver must first be granted by the TSC and the exception documented in the release notes for that specific ONAP module.

Creating a Framework for Developing Secure Software

OpenSFF Security Best Practices Badging

The Open Source Security Foundation (OpenSSF) Best Practices badge (formerly the “Core Infrastructure Initiative badge”) is a way for Free/Libre and Open Source Software (FLOSS) projects to show that they follow best practices. Projects can voluntarily self-certify, at no cost, by using this web application to explain how they follow each best practice. The OpenSSF Best Practices Badge is inspired by the many badges available to projects on GitHub. Consumers of the badge can quickly assess which FLOSS projects are following best practices and as a result are more likely to produce higher-quality secure software.

LFN projects aim to achieve the highest level of security badges, and our communities have been working hard to reach targeted goals. Currently, all maintained ONAP sub-projects have earned

"Passing" level badges, with several sub-projects achieving "Silver" level status (a list of them can be found here), and other LFN projects including EMCO, OpenDaylight, and Tungsten Fabric are working through the process.

OpenSSF Badging Levels

There are three OpenSSF Badging levels which are as follows: Passing, Silver, and Gold.

When a new project starts the badging process they will begin at 0 percent completeness, and as they progress, the percentage will increase.

Level	Example Details/Criteria
Passing	The project website MUST succinctly describe what the software does (what problem does it solve?). 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).
Silver	The project MUST document what the user can and cannot expect in terms of security from the software produced by the project. The project MUST identify the security requirements that the software is intended to meet and an assurance case that justifies why these requirements are met. The assurance case MUST include: a description of the threat model, clear identification of trust boundaries, and evidence that common security weaknesses have been countered
Gold	The project MUST have at least 50% of all proposed modifications reviewed before release by a person other than the author, to determine if it is a worthwhile modification and free of known issues which would argue against its inclusion.

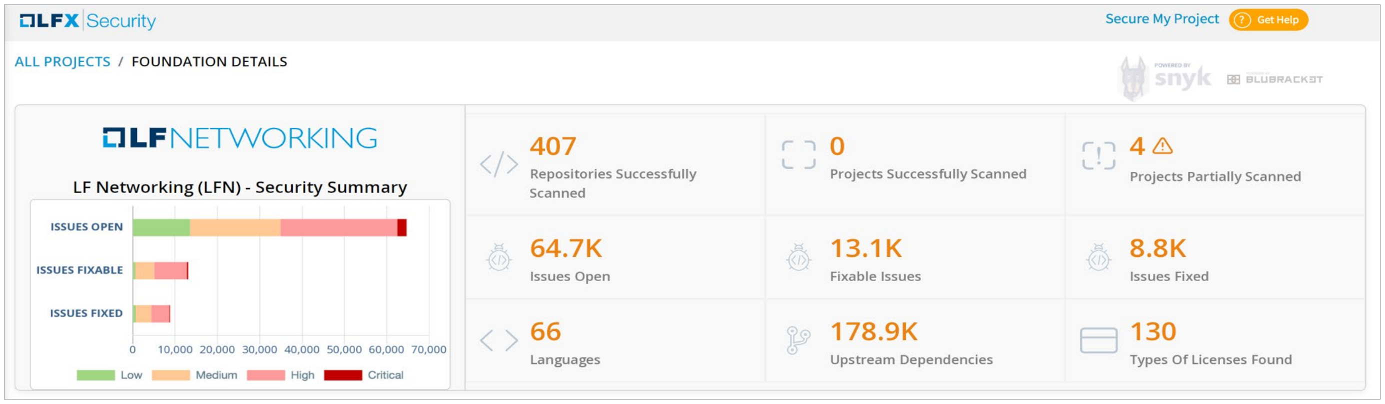
Criteria for "passing" are available here.

OpenSSF badge status of LFN projects:

- **Anuket**
  - The Open P:latform for Network Functions Virtualization (OPNFV) project, which is folded into the Anuket project, has earned 115% "passing" OpenSSF badging status, making Anuket more secure out of the gate.
- **ONAP**
  - Currently, 38 ONAP projects have earned a "passing" or above OpenSSF bading status.
  - This includes: CLAMP; ONAP AAI UI; CLI, logging-analytics; ONAP VNFSDK; ONAP Active and Available Inventory; Microservices Bus (MSB); ONAP Holmes; ONAP External System Register (ESR); Virtual Function Controller (VF-C); ONAP Policy; ONAP Service Design and Creation (SDC); ONAP Common Controller SDK (CCSDK); ONAP Virtual Infrastructure Deployment (VID);ONAP SO; ONAP SDN Controller; Multi-VIM/Cloud; ONAP Data Collection Analytics and Events (DACE); ONAP Optimization Framework (OOF); MUSIC; ONAP AAI UI; VVP; ONAP Model Loader; ONAP Babel; ONAP-DMaaP-messagerouter; Usecase UI; ONAP External API; ONAP Modeling; ONAP AAI Rest Client; and ONAP Champ.
  - Four additional projects are in progress towards earning a "passing" or above status. More details on ONAP OpenSSF bading progress by project are available here: ONAP OSSF badging status
  - ONAP has become more secure with each release. See the current list of ONAP OSSF badging progress by Release, for more details.
- **EMCO**
  - Pending work item/Work on the horizon
- **L3AF**
  - Pending work item/Work on the horizon

Tracking and Managing Vulnerabilities

LFX Security Dashboard



407 Repositories Successfully Scanned	0 Projects Successfully Scanned	4 ⚠ Projects Partially Scanned
64.7K Issues Open	13.1K Fixable Issues	8.8K Issues Fixed
66 Languages	178.9K Upstream Dependencies	130 Types Of Licenses Found

LFX Security, which is one tool of the LFX platform—built to facilitate every aspect of open source development, including health monitoring of projects and communities, operations maintenance, and growth potential with contributor tools— provides a clear view into the security of a given project and enables developers to identify and resolve vulnerabilities quickly and easily. LFN projects are increasingly relying on LFX to enable identification and resolution of vulnerabilities. The program's features include:

- **Code Secrets:** Find and fix any non-public information that may be exposed in the code
- **Non- Inclusive Language Scan:** Address any imbalances and discrimination inherent in existing language
- **Updated Dependency Tree:** Now view the code dependencies by priority, location and which licenses they are coming from
- **Automated Vulnerability Scanning:** Monitor the health status against thousands of authorized open source repository vulnerability databases, bug bounties, security advisories, and security reports
- **Neutral to Source Control Systems:** Security supports the most common SCMs including GitHub, Bitbucket, GitLab, Azure, and more
- **License Compliance Management:** Ensure project compliance by keeping track of all licenses used by your projects and their dependencies

LFN projects are among the first to harness the power of LFX security, with ONAP currently the most active project in both "vulnerabilities fixed" and "repositories scanned" categories.

Software Bill of Materials (SBOM)

What is an SBOM?

An SBOM is a formal, machine-readable machine-readable list of an application's software components and their dependencies. Each component is represented by machine-readable metadata that uniquely identifies a software package and its contents; it may include other information about its contents, including copyrights and license data. SBOMs are designed to be shared across organizations and are particularly helpful at providing transparency of components delivered by participants in a software supply chain.
Many organizations concerned about software security are making SBOMs a cornerstone of their cybersecurity strategy. In July 2021, the National Telecommunications and Information Administration (NTIA) of the US Department of Commerce issued recommendations for SBOMs, providing a de facto standard.

Given that software is made of bits of code and subroutines from numerous sources, the SBOM is a list of all components, meta data and relationships.

Security-specific SBOMs

The Linux Foundation recently issued "The State of Software Bill of Materials and Cybersecurity Readiness," which reports on the extent of organizational SBOM readiness and adoption tied to cybersecurity efforts across the open source ecosystem. The study comes on the heels of the US Administration's Executive Order on Improving the Nation's Cybersecurity, and the disclosure of the most recent and far-reaching log4j security vulnerability. Its timing coincides with increasing recognition across the globe of the importance of identifying software components and helping accelerate widespread implementation of cybersecurity best practices to mitigate the impact of software vulnerabilities. The report is a good reference for any organization looking to implement SBOMs to aid in security readiness.

"An SBOM is a formal, machine-readable machinereadable list of an application's software components and their dependencies."

Benefits to implementing a security SBOM include:

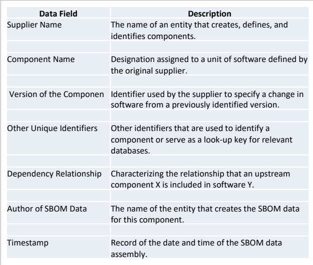
Transparency (Software Supply Chain)

- Cyber Risk Management-
  - CTI (Cyber Threat Intelligence) to assets mapping
  - Assets to control mapping
- Source code & license management
- Early identification and mitigation of vulnerable systems
- Compliance and reporting (i.e., Gov. requirement for Software Composition Analysis-SCA, EO 14028)

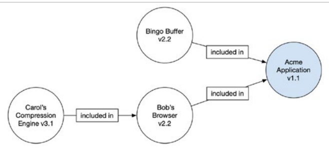
What’s in the SBOM?

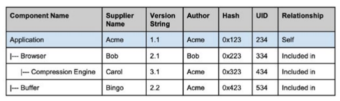
Recent recommendations from the the NITA, based on industry feedback suggest it include:

- Suppliers name
- Component name
- Component version
- Other unique identifiers
- Dependency relationship
- Author
- Timestamp
- Nesting capabilities



Data Field	Description
Supplier Name	The name of an entity that creates, defines, and identifies components.
Component Name	Designation assigned to a unit of software defined by the original supplier.
Version of the Componen	Identifier used by the supplier to specify a change in software from a previously identified version.
Other Unique Identifiers	Other identifiers that are used to identify a component or serve as a look-up key for relevant databases.
Dependency Relationship	Characterizing the relationship that an upstream component X is included in software Y.
Author of SBOM Data	The name of the entity that creates the SBOM data for this component.
Timestamp	Record of the date and time of the SBOM data assembly.





Component Name	Supplier Name	Version String	Author	Hash	UID	Relationship
Application	Acme	1.1	Acme	0x123	234	Self
|--- Browser	Bob	2.1	Bob	0x223	334	Included in
|--- Compression Engine	Carol	3.1	Acme	0x323	434	Included in
|--- Buffer	Bingo	2.2	Acme	0x423	534	Included in

Source: www.ntia.gov, note minimum data field shown in the table, NTIA report also presents expanded data fields

There are three ISO-recommended formats available, including:

- **SPDX** (adopted as ISO standard) is an open standard for communicating software bill of material information (including components, licenses, copyrights, and security references). The SPDX specification is developed by the SPDX workgroup, which is hosted by the Linux Foundation. The grass-roots effort includes representatives from more than 20 organizations—software, systems and tool vendors, foundations and system integrators.
- **Cyclone DX** is a software bill of material (SBOM) standard, purpose-built for software security contexts and supply chain component analysis. The specification is maintained by the Cyclone DX Core working group, with origins in the OWASP community.
- **SWID tags** record unique information about an installed application, including its name, edition, version, whether it is part of a bundle and more. SWID tags support software inventory and asset management initiatives. The structure of SWID tag is specified in international standard ISO/IEC 19770-2:2015.

ONAP’s Security SBOM pilot

As discussed previously, an SBOM is a formal, machine-readable inventory of software components and dependencies, information about those components, and their hierarchical relationships. There are several benefits of creating and using an SBOM, including reduction of costs, security risks, license risks, and compliance risks. Overall, SBOMs help improve software development, supply chain management, vulnerability management, asset management, procurement, and high- assurance processes.

As one of the most mature and widely-deployed LF Networking projects, the ONAP community implemented a pilot SBOM. The template — which includes SPDX as the selected ISO standard and implementation of a real ONAP SBOM within the LF Networking CI pipeline — can be easily leveraged by other open source groups.

SPDX - SBOM File Header

When running a BOM function in a pipeline, it's usually when ready to put a release out into repo, with no more action required beyond that of the software development lifecycle. You'll run an SBOM text file that will be generated by running the tool on code — either code, binary, or container.

```
SPDXVersion: SPDX-2.2
DataLicense: CC0-1.0
SPDXID: SPDXRef-DOCUMENT
DocumentName:so-1.9.0-SNAPSHOT
Created: 2021-10-13T03:08:25Z
```

In this case, we used an SO, and can see a snapshot of the software:

- At the top, it tells you which version you are using. Here we're using SPDX, which is licensed under Creative Commons so the user owns the data and the documentation. No vendor/produce tie-back to the technology.
- It shows the namespace (which can be unique), creator field (which can be a tool, an individual, or an organization).
  - In this example, the creator is a tool because we're using an automated system. But this can be created manually as well.

SPDX - Package Information

Here's a package representation for SO scan, where we're using SO as an organization (but this can be configured to show LF Networking ONAP as the organization as well).

```
PackageName: so
SPDXID: SPDXRef-Package-so
PackageDownloadLocation: https://mvnrepository.com/artifact/org.onap.so
FilesAnalyzed: false
PackageHomePage: NOASSERTION
PackageLicenseConcluded: NOASSERTION
PackageLicenseComments: NOASSERTION
PackageComment: NOASSERTION
```

Then there's a download location, which indicates if you are delivering through web delivery or FTP delivery.

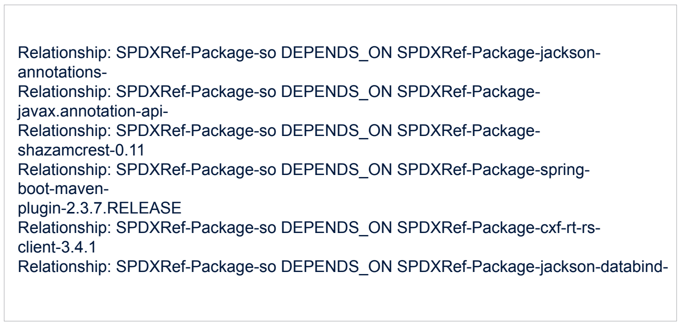
There's CHECKSUM — by default it's SHA1, but there are four other options you can choose to add CHECKSUM to.

There are options to include other metadata: here you see new insertion, SPDM flex, etc. (if you cannot get that information, or purposely choose not to include it, it can be presented as "NO ASSERTION").

"A lot of what is built in [SPDX] currently addresses needs of developers, consumers and security risk managers."

Relationship Roadmap

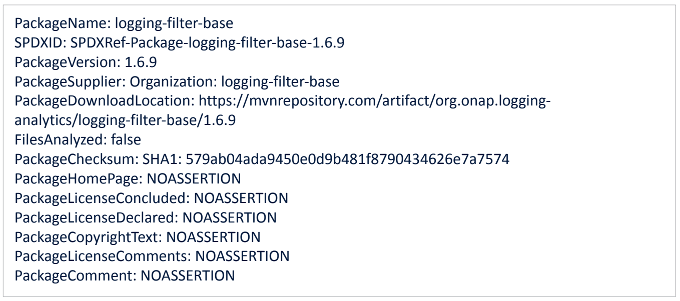
Here is a relationship map.



```
Relationship: SPDXRef-Package-so DEPENDS_ON SPDXRef-Package-jackson-
annotations-
Relationship: SPDXRef-Package-so DEPENDS_ON SPDXRef-Package-
javax.annotation-api-
Relationship: SPDXRef-Package-so DEPENDS_ON SPDXRef-Package-
shazamcrest-0.11
Relationship: SPDXRef-Package-so DEPENDS_ON SPDXRef-Package-spring-
boot-maven-
plugin-2.3.7.RELEASE
Relationship: SPDXRef-Package-so DEPENDS_ON SPDXRef-Package-cxf-rt-rs-
client-3.4.1
Relationship: SPDXRef-Package-so DEPENDS_ON SPDXRef-Package-jackson-databind-
```

It will provide a relationship map that the SO package has a bunch of other packages that make up the SO application, which are listed with the correct corresponding versions.

Included Packages



```
PackageName: logging-filter-base
SPDXID: SPDXRef-Package-logging-filter-base-1.6.9
PackageVersion: 1.6.9
PackageSupplier: Organization: logging-filter-base
PackageDownloadLocation: https://mvnrepository.com/artifact/org.onap.logging-
analytics/logging-filter-base/1.6.9
FilesAnalyzed: false
PackageChecksum: SHA1: 579ab04ada9450e0d9b481f8790434626e7a7574
PackageHomePage: NOASSERTION
PackageLicenseConcluded: NOASSERTION
PackageLicenseDeclared: NOASSERTION
PackageCopyrightText: NOASSERTION
PackageLicenseComments: NOASSERTION
PackageComment: NOASSERTION
```

Then each package itself will have similar information to what's included in the relationship map e.g., package name, version, where to download, licensee, etc.

- You can use a concluded or inherited license — if you use another product that's licensed, and you assume the licensee's responsibility, you can note "included" or "inherited," or in some cases.

When comparing what we get out of using SPDX and NTIA's minimum recommended fields, we are meeting these requirements. Part of the initial recommendation emphasizes that right now is a good time to start, organizations should not hold off on creating an SBOM until there is an extended BOM available.

What's next?

- Review SPDX documentation
  - SPECS and SBOM generator tool
- Select a pilot project
  - Plan dates
  - Engage LFN Release Engineering
    - Will help you install script into your pipeline
    - No maintenance once you fix your CI task, it's all automated

While SPDX might be updated in the future (e.g. SPDX version 3 may contain additional functions and capabilities), a lot of what is built in currently addresses needs of developers, consumers and security risk managers.

- This example/pilot was created and built as a new repo, outside of the Linux Foundation — it was downloaded, created and run as a tool in a new environment. It's automated and designed to be easy to use so the only thing that's needed is to add an SPDX SBOM file.

Additional useful references:

- SPDX V2.2.1 Specification
- SPDX Online Tools
- NTIA SBOM Overview
- ONAP SBOM Wiki
- LFN-SCANs for ONAP

Other LFN projects are adopting their own SBOM. Specific examples in progress include:

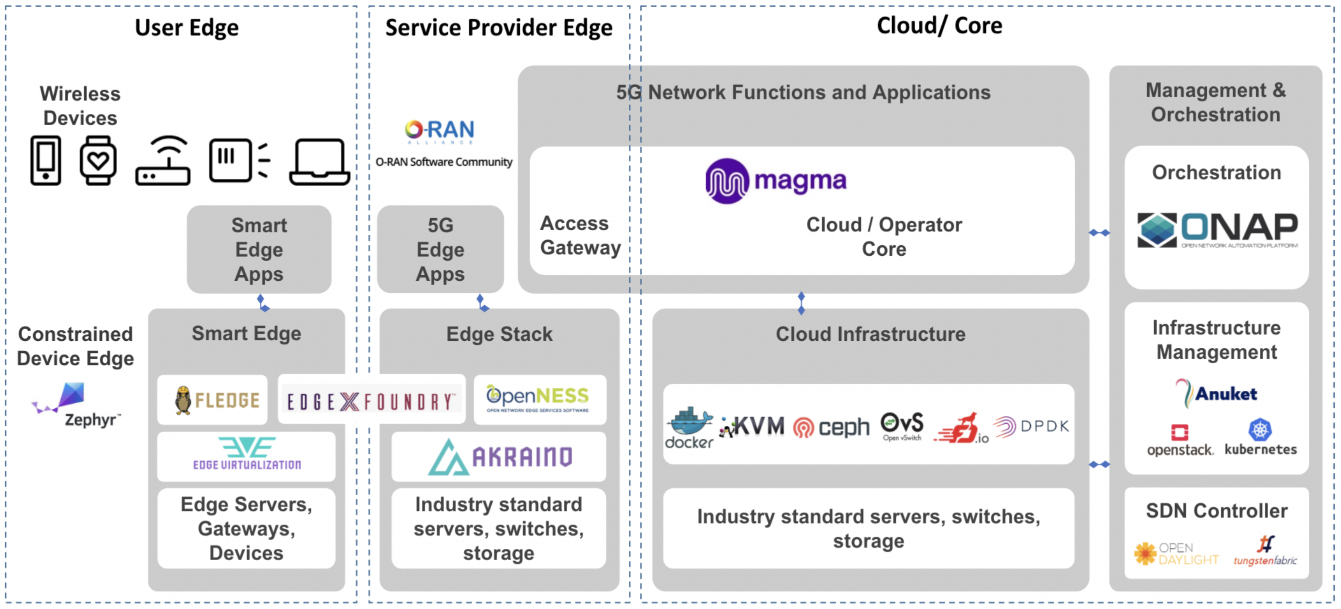
- **Anuket** - The project identified SBOM as one of the security measures for creating robust Telco infrastructure. It is already included as a part of the project's holistic security requirements specifications. Future releases of the projects will continue to add more details regarding the format of the SBOM.

- **OpenDayLight** - The community just recently successfully added automatic generation of SBOM using the CycloneDX format. This is expected to become available in the coming Sulfur release and Phosphorus Service Release 3.

The 5G Super Blue Print - Securing 5G from End to End, with LF Networking & Beyond

As global communications providers prepare to deliver high-speed connectivity to support new services and use cases, the need for low-latency, high-bandwidth, scalable networks is more important than ever — as is the security of those networks. Software-defined infrastructure and cloud native approaches are essential for delivering the performance, capabilities, and automation 5G requires. This software defined approach exposes networks to new threats. The security measures described in this document are aimed at addressing such threats.

LF Networking is leading a community-driven integration and proof of concept — the "5G Super Blue Print" — involving multiple open source initiatives to demonstrate end-to-end implementation architectures for end users covering RAN, Edge, and Core. As seen in the diagram below, there are many pieces to this puzzle. The end to end security of this architecture is only as strong as the weakest link in the chain. Therefore the importance of security is paramount to all involved open source projects. Network architects and security professionals need a way to map and trace any vulnerabilities in the software used and its dependencies. Tools like SBOM and OpenSSF badging could provide the required peace of mind to these individuals.



Closing

We are living in times of ever increasing cyber security threats. Geo political instability gives rise to new types of malicious attacks. Increased use of open source software for our critical infrastructures makes its security aspects more important than ever. We cannot afford to apply software security as an afterthought. As shown in the examples in this document, robust security is a holistic approach, covering all phases of software development and deployment. With the proliferation of open source software and increased threats, our work is just beginning. For a chance to stay ahead, we need the collaborative effort of experts and enthusiasts from anywhere in the software development industry.

The Linux Foundation Networking has several established platforms for collaboration. They are all open to the public, and welcome contributions in any form or shape. Please join the conversation in our LFN Security Forum, or any other of the project-specific committees and help us by:

- Sharing your security concerns, requirements and identified threats.
- Contributing best practices and recommendations.
- Analyzing our projects' software and reporting vulnerabilities.
- Propose enhancements and improvements to projects.
- Contribute open source code to improve our projects.

There are certainly more ways in which you could help. Everyone is welcome to join the discussion even if they are not sure yet how to contribute. We have a long journey ahead and opportunities are bound to come up for all sorts of contributions.