Supply Chain Security Assessment of Container Images in Automated 5G/6G Deployments

David Company
i2CAT Foundation, Cybersecurity Dept.
Barcelona, Spain
david.company@i2cat.net

Maxime Compastié
i2CAT Foundation, Cybersecurity Dept.
Barcelona, Spain
maxime.compastie@i2cat.net

***Abstract*—The involvement of open-source communities and the adoption of cloud-native technologies are key trends in the realization of next-generation mobile networks (6G), but they widen the exposure of downstream 6G appliances to software supply chain attack vectors. This paper studies the exposure of the open-source software ecosystem expected to underpin the 6G software fabric, identifying prominent threats. Our methodology selects key open-source communities involved in 6G development and systematically analyzes the software artifacts they deliver against widely adopted software supply chain security benchmarks. We propose an implementation prototype of this methodology to enact trust evaluation of supply-chain of networks appliances. We synthesize our findings into recommendations for academics, standardization bodies, and R&D communities to bolster the security posture of their development and delivery processes.**

***Index Terms*—6G, software supply-chain, trust management, zero-trust, open source, cloud native**

I. Introduction

The next generation of mobile networks (6G) is anticipated to involve software at the very heart of its capability: Beyond inheriting the adoption network softwarization and cloud native technologies from 5G, the exposure of network capabilities via application programming interface (e.g. CAMARA and OpenCAPIF projects [1]) will facilitate application developers to access contextual network information and implement the specific requirements of verticals. In this sense, several open-source software (OSS) communities have already established themselves as key providers of network technologies [2], covering functionalities from core networks to radio access networks.

The emerging role of these communities comes, however, with the downside of complicating the supply chain of network technologies, as each contributing communities implement their own governance, software engineering and software development lifecycle (SDLC) processes and tooling. As the software supply chain have become increasing subjected to compromises, it has only become natural for these communities to become attractive targets, thus opening a new threat surface in telecommunication platforms. Given the regulation momentum for telecommunication resilience [3], [4], there is a strong incentive to evaluate trust in these communities based on their ability to deliver secured and tampered-free software artifacts used in network operation.

The objective of this paper is to evaluate the current degree of preparedness of open source communities, in regards to their involvement in enabling technologies and features, by answering the following questions:

- RQ1 — To what extent do the OSS container images underpinning automated 5G/6G deployments adopt each of the SDLC supply-chain controls?
- RQ2 — How does supply-chain maturity vary across the functional categories of the 6G cloud-native stack?
- RQ3 — What actionable recommendations for 5G/6G platform operators, project maintainers, and the research community can be derived from the observed supply chain security posture and its recurring adoption patterns?

To that end, the contributions of this paper include (i) the proposition of a trust analysis methodology for software container images delivered by OSS communities (ii) the preparation of STELA-6G (Supply-chain Trust and Ecosystem Lifecycle Assessment), a tool for implementing this analysis (iii) a set of recommendations for stakeholders to bolstering the security posture of their supply chain. This study is based on a selected set of OSS communities involved 5G deployment, but anticipated to actively involved in the design of of 6G-specific functionalities.The supplementary material supporting the study and recommendations of this paper is available in an open repository [5].

The remainder of this paper is organized as follows: Section II provides an introduction to the security of software supply-chain and present related work. Section III describes and justify the methodology adopted by our methodology while the implementation prototype implementing this methodology is presented in Section IV. The results of our study are disclosed in Section V and we draw recommendations in Section VI. Section VII concludes this paper.

II. Background & Related Work

A. Securing the Software Supply Chain: Threats and Defences

OSS supply chain attacks have grown substantially in recent years. Following the taxonomy proposed by Ladisa et al. [6], such attacks can be broadly classified into three categories: the creation of malicious packages disguised as legitimate software; name-confusion attacks, such as typo-squatting, that lead developers and operators to unknowingly include a compromised package in their dependency tree; and the subversion of

979-8-3195-0167-7/26/$31.00 ©2026 IEEE

an already legitimate package. Williams et al. [7] identify the mechanisms that most commonly enable this subversion: the compromise of build infrastructure, such as CI/CD pipelines, and the exploitation of the human factor through social engineering directed at maintainers and contributors. This reliance on third-party components, and on the trust relationships that bind together modern software ecosystems, creates a broad and largely opaque attack surface for adversaries to exploit. The 2020 SolarWinds compromise illustrates this risk: attackers infiltrated the build pipeline of the Orion platform and injected malicious code that became indistinguishable from legitimate functionality once compiled [7].

In response to this growing threat, the community has developed tools and practices aimed at increasing the visibility and transparency of the software development process, most notably under the DevSecOps paradigm, which embeds security controls throughout the development pipeline rather than applying them as a late-stage gate — including Software Bills of Materials (SBOM, in SPDX or CycloneDX format), SLSA (Supply-chain Levels for Software Artifacts) provenance attestations, and the cryptographic signing of artifacts prior to deployment. Several prominent international bodies have published guidance and best practices standardizing these defenses. The CNCF (Cloud Native Computing Foundation) reference architecture [8] for a secure software factory, for instance, frames supply chain security as spanning four complementary stages: hardening the source repository, vetting ingested third-party materials, securing the build pipeline, and attaching signed, independently verifiable metadata to the resulting artifact — closely mirroring the four dimensions evaluated in this study. ENISA (European Union Agency for Cybersecurity) [9], CISA (Cybersecurity and Infrastructure Security Agency), and NIST (National Institute of Standards and Technology) [10] have published complementary guidance addressing similar concerns.

B. Empirical Evaluation and Trust Scoring of Supply-Chain Posture

Several empirical studies have measured the adoption of supply chain security practices — such as signing and SLSA provenance — across open-source ecosystems. Zahan et al. [11] study the npm and PyPI ecosystems and identify several weak link signals — indicators derived from public package metadata that correlate with a heightened risk of compromise. They argue that npm packages currently lack any overall security or health score, making it difficult for developers to assess the risk of adopting a new dependency, and propose incorporating these weak link signals into existing community efforts such as OpenSSF Metrics and OpenSSF Scorecard. Kalu et al. [12] investigate the industrial adoption of software signing and identify the limited verification of digital signatures as a recurring obstacle, proposing signed metadata as a means of improving verification processes. Mills et al. [13] assess risk in Docker container images through a longitudinal study, showing that vulnerability exposure tends to increase over time and arguing for context-aware risk assessment that incorporates exploitability indicators such as EPSS (Exploit Prediction Scoring System) and KEV (Known Exploited Vulnerabilities) alongside raw vulnerability counts.

Beyond these adoption studies, scoring frameworks have been proposed to quantify supply chain security posture at scale. The OpenSSF Scorecard [14], discussed above, evaluates a source repository against a fixed set of best-practice checks and returns a single automated score. Enck et al. [15], drawing on interviews with 30 industry and government organizations, identify five recurring supply chain security challenges: managing vulnerable dependencies, leveraging SBOMs effectively, selecting trustworthy dependencies, assuring the build process, and achieving industry-wide participation in these practices — challenges that map closely onto the four dimensions evaluated in this study. However, to our knowledge, no prior work applies an integrated, multi-dimensional scoring framework specifically to the open-source image ecosystem underpinning 5G/6G deployments — jointly assessing source repository posture, third-party dependency exposure weighted by exploitability (EPSS/KEV), artifact signature verification, and build environment trustworthiness via the SLSA standard. This focus on container images, rather than on the development-time dependency packages studied by much of the work discussed above, reflects an emphasis on the artifact that is actually deployed: in cloud-native 5G/6G infrastructure, admission and trust decisions are made at the image level, not at the level of individual libraries incorporated during development. This is the gap addressed by the present study.

C. Supply-Chain Security in Next-Generation Mobile Networks

5G/6G cloud-native networks are deployed as containerized network functions (CNFs) on Kubernetes infrastructure, which transposes software supply chain risk onto the network plane itself. Chen et al. [16] articulate a zero-trust architecture for 6G networks, noting that 6G will rely extensively on open-source software to implement network functions, which exposes operators to the exploitation of software vulnerabilities, including zero-day exploits and malware. In the context of software supply chain security, zero trust entails verifying — prior to deployment — that a given container image is the correct, intended artifact; that it was developed and built under demonstrably secure conditions; and that it is free of known critical third-party vulnerabilities, rather than extending implicit trust based on an artifact's origin, registry, or apparent provenance. Reed et al. [17] provide a broader 5G/6G security survey that frames this architectural shift — toward NFV (Network Functions Virtualization)- and SDN-based, software-defined infrastructure and a wider reliance on open-source components and multi-vendor diversity — as a driver for adopting zero-trust principles across the network stack. The automation of the deployment life-cycle, through GitOps pipelines and Kubernetes admission control, means that a compromised or unverified image can propagate into production network infrastructure with minimal human oversight. Unlike generic cloud-native ecosystems studied within

TABLE I
CORPUS COMPOSITION — 23 PROJECTS, 105 IMAGES

Project	Category	Images	Registry
Nephio	5G/6G Core NFs	5	Docker Hub
SD-Core	5G/6G Core NFs	10	Docker Hub
free5GC	5G/6G Core NFs	12	Docker Hub
OpenAirInterface	5G/6G Core NFs	8	Docker Hub
srsRAN Project	RAN & RIC	1	Docker Hub
UERANSIM	RAN & RIC	2	Docker Hub
ONOS SD-RAN	RAN & RIC	8	Docker Hub
KubeEdge	Edge / MEC	4	Docker Hub
EdgeX Foundry	Edge / MEC	7	Docker Hub
Cert-manager	Zero-trust & PKI	3	Quay.io
Cilium	Zero-trust & PKI	5	Quay.io
SPIFFE-SPIRE	Zero-trust & PKI	5	GHCR
Istio	Zero-trust & PKI	5	Docker Hub
MetalLB	CN Networking	2	Quay.io
OpenTelemetry	CN Networking	3	Docker Hub
Telco CNI	CN Networking	3	GHCR
Kube-OVN	CN Networking	1	Docker Hub
SR-IOV Net. Operator	CN Networking	3	GHCR
Argo CD	GitOps / CD	1	Quay.io
Flux	GitOps / CD	6	GHCR
Argo Rollouts	GitOps / CD	1	Quay.io
OSM	NFV Orchestration	6	Docker Hub
ONAP	NFV Orchestration	4	Docker Hub

the CNCF community, the open-source 5G/6G image ecosystem has not been the subject of a systematic assessment of its supply chain security posture, despite the fact that the projects supplying these images are intended for automated network deployments where trustworthiness cannot be assumed. The NIS2 Directive [3] and the emerging Cyber Resilience Act [4] add a regulatory compliance dimension to this need, further motivating the empirical characterization undertaken in this paper.

III. METHODOLOGY

A. 6G Cloud-Native Architecture and Project Selection

The container images constituting the project corpus were selected by mapping open-source projects to functional roles in the 6G cloud-native deployment stack. As detailed in Table I, the stack comprises seven functional layers: the *GitOps/continuous-delivery* substrate (Flux, Argo CD, Argo Rollouts), responsible for the automated, declarative deployment of network infrastructure; the *NFV orchestration layer* (OSM, ONAP), managing the lifecycle of virtualized network functions; *5G/6G core network functions* (Nephio, OpenAirInterface, free5GC, SD-Core), implementing the 3GPP service-based architecture; *RAN and RIC* components (ONOS SD-RAN, srsRAN, UERANSIM), covering the radio access and intelligent RAN controller tiers; the *edge and MEC layer* (KubeEdge, EdgeX Foundry), enabling distributed processing at the network edge; cloud-native networking infrastructure (Cilium, Telco CNI, MetalLB, Kube-OVN), providing CNI, load-balancing, and high-performance data-plane capabilities; and a *zero-trust security and identity layer* (Istio, SPIFFE-SPIRE), enforcing mutual TLS and workload identity across the deployment.

B. Corpus Selection Criteria and Composition

Project and image inclusion was governed by six criteria. C1 requires that each project fills a defined functional role in the 6G cloud-native deployment stack, as described in Section III-A. C2 requires that container images be published directly by the project developers, ensuring that the SDLC (Software Development Lifecycle) chain evaluated by this study is attributable to the originating team. C3 requires pinned, semantically versioned release tags; images published exclusively under generic references such as :latest are excluded. C4 requires evidence of active maintenance, assessed through recent release activity and community engagement. C5 requires that images be publicly accessible without authentication. C6 excludes images whose deployment role is limited to CI pipelines, testing harnesses, or auxiliary helpers with no independent production role.

Table I summarizes the corpus composition: 105 container images drawn from 23 open-source projects, distributed across the seven functional categories described in Section III-A. The number of images per project ranges from 1, for projects with a narrowly scoped deployment role, to 12, for projects covering a broader set of network functions. Together, the corpus provides coverage of a representative generic 6G cloud-native deployment stack. Several otherwise prominent projects — including O-RAN SC and TeraFlowSDN — were not incorporated, as they did not satisfy all selection criteria.

IV. METHODOLOGY IMPLEMENTATION: STELA-6G

A. Framework Architecture and Rationale

STELA-6G is a prototype tool developed within the iTrust6G project to operationalize the static supply chain trust assessment methodology defined in deliverable D4.1 [18]. It evaluates a container image prior to deployment by checking for the presence of established security mitigations across four phases of the software supply chain — source, build, third-party dependencies, and artifact distribution — as established by the CNCF secure software factory reference architecture [8] and corroborated by industry practice in a recent interview study of recurring supply chain security challenges [15]. STELA-6G produces a per-dimension supply chain posture score and a composite score, both in the range [0, 1], where 1 denotes full trust and 0 denotes the absence of verifiable mitigations. This pre-deployment, evidence-based assessment, depicted in Fig. 1, is designed to inform — rather than automatically enact — admission decisions in automated 6G environments, where runtime inspection is impractical.

B. The Four Dimensions of the Supply Chain Analysis

Building on the supply chain framework and reference architecture introduced above, measuring the trustworthiness of an artifact prior to deployment requires analyzing and verifying security independently across each of these four phases, or dimensions, of the supply chain.

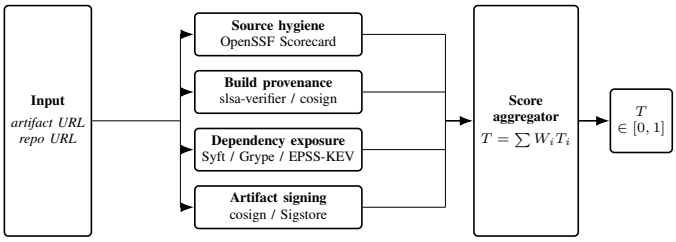


Fig. 1. STELA-6G evaluation pipeline: four dimension analyzers feeding a weighted score aggregator.

a) Source hygiene: At the source phase, the principal threat is the introduction of malicious code or vulnerabilities into the source repository — whether through a compromised contributor account or inadequate development controls. The corresponding mitigations are secure development practices and the hardening of the source repository. STELA-6G integrates the OpenSSF Scorecard [14], which validates the presence of these mitigations by performing a defined set of checks against the project's source repository URL, evaluating a well-established list of security best practices: branch protection policies, verified merge procedures, multi-organization contributor participation, code review enforcement, and the absence of executable binaries in the repository, among others. Scorecard returns a score from 0 to 10, which STELA-6G normalizes to [0, 1].

b) Build provenance: At the build phase, the principal threat is malicious code injection — an attacker compromising the build toolchain or modifying build scripts outside the normal approval cycle. The corresponding mitigation is a secure build process: an isolated, immutable, and preferably ephemeral build environment, as prescribed by the SLSA framework. A cryptographically signed provenance attestation — stored as an OCI (Open Container Initiative) attestation alongside the container image in the registry — serves as verifiable evidence that these conditions were met during the build. STELA-6G verifies adherence to SLSA, which defines four integrity levels from L0 (no controls) to L3 (a hardened, isolated, ephemeral build environment with verifiable provenance), either using slsa-verifier — for instance, when the build leverages slsa-github-generator, a toolset for generating SLSA Level 3 provenance for native GitHub projects — or using cosign to verify the attestation signature and validate the extracted provenance fields against a pre-configured root of trust that establishes the accepted builder identities and expected provenance values, such as the builder ID and the SLSA level, ensuring that only attestations issued by trusted builders are accepted; the root-of-trust configuration used in this evaluation is published in the companion Zenodo dataset.

c) Dependency exposure: At the dependency phase, the threat is that reused third-party libraries or components introduce critical vulnerabilities into an otherwise secure artifact. STELA-6G generates a Software Bill of Materials (SBOM) using Syft and then scans it with Grype to identify known vulnerabilities. The resulting findings are post-processed using the EPSS and KEV indicators to distinguish vulnerabilities with high exploitation likelihood from those that are theoretical. The dimension score equals one minus the EPSS-weighted average risk across dependency vulnerabilities above the 30th EPSS percentile, where each vulnerability's risk combines CVSS-based severity with an exploitation-likelihood factor elevated for KEV-listed vulnerabilities; a score of 1 indicates the absence of such significant, actively exploited vulnerabilities.

d) Artifact signing: At the distribution phase, the threat is substitution or tampering of the artifact in the registry — including typosquatting attacks. The corresponding mitigation is a cryptographic signature attached to the artifact, which allows its authenticity and integrity to be proven prior to deployment. STELA-6G uses cosign (Sigstore framework) to verify that the image carries a valid signature issued by a trusted identity, checked against a pre-configured root of trust that defines the accepted signing authorities and certificate issuers. A score of 1 indicates that a signature is present and has been successfully verified.

C. Composite Scoring Model

The four dimension scores are aggregated into a single composite posture indicator using a weighted sum:

$$T_{\text{final}} = \sum_{i=1}^{4} W_i \cdot T_i, \quad \sum_{i=1}^{4} W_i = 1, \quad W_i \geq 0 \qquad (1)$$

where $T_i \in [0, 1]$ is the posture score for dimension i and W_i is its assigned weight. By default, equal weights are applied ($W_i = 1/4$), reflecting a conservative assumption in the absence of a deployment-specific risk profile. The weights are policy-configurable to reflect operator priorities; alternative risk-weighted profiles are analyzed in Section VI-D. When a dimension score cannot be computed — for instance, due to an inaccessible registry or an incompatible toolchain — that dimension is excluded from the aggregation and the remaining weights are renormalized, so that the composite score always reflects the checks that were actually performed. This exclusion applies only when the evaluation tool itself cannot execute a check due to environment incompatibility; a dimension that is measurable but shows no control in place — such as absent build provenance or artifact signing — is never renormalized away and always scores 0.

V. Evaluation & Results

A. Evaluation Protocol

The analysis was conducted using the following tool versions: cosign v3.0.5, OpenSSF Scorecard v5.0.0, slsa-verifier v2.7.0, Syft v1.44.0, and Grype v0.112.0. Execution ran on a commodity Linux workstation (Intel Core Ultra 5 235U, 16 GB RAM); the computational footprint is modest, as the analysis is I/O-bound rather than compute-intensive. STELA-6G exposes a FastAPI endpoint that orchestrates the per-phase Python analysis scripts; corpus entries are processed sequentially, and results are consolidated into a structured JSON

TABLE II
MEAN SCORES BY FUNCTIONAL CATEGORY

Category	Source	Build	Deps	Artifact	Global
A · 5G/6G Core NFs	0.593	0.000	0.956	0.114	0.416
B · RAN & RIC	0.355	0.000	0.202	0.000	0.139
C · Edge / MEC	0.717	0.273	0.915	0.000	0.476
D · Zero-trust & PKI	0.702	0.000	0.798	0.889	0.597
E · CN Networking	0.657	0.000	0.896	0.333	0.471
F · GitOps / CD	0.829	1.000	0.810	1.000	0.910
G · NFV Orchestration	0.358	0.000	0.566	0.000	0.231
Corpus mean	**0.603**	**0.105**	**0.791**	**0.305**	**0.451**

file recording per-image dimension scores and the final composite score. Mean analysis time per image is approximately 3–4 minutes, with SBOM generation and Grype vulnerability scanning representing the dominant cost. Authenticated access to the GitHub API and Docker Hub was required to avoid rate-limiting constraints; the corpus was processed in batches accordingly.

B. Aggregate Results

The evaluation covers the full corpus described in Section III-B. Per-project mean scores for all 23 projects, together with per-image evaluation timestamps, are provided as a companion dataset in a public Zenodo repository [5], given space constraints. As a robustness check using this per-project data, category-level means recomputed with each project weighted equally, rather than by image count, leave category rankings and the GitOps/CD–RAN & RIC gap substantially unchanged, confirming the reported findings are not an artifact of the corpus's uneven per-project image counts. Scores range from 0 to 1 per dimension, where 1 denotes full adoption of the corresponding verified supply chain controls. As shown in Table II, the mean source score across the corpus is 0.603; the mean build score is the lowest at 0.105; the mean dependency score is 0.791; and the mean artifact score is 0.305, yielding a corpus-wide composite mean of 0.451. Notably, 66.7% of the analyzed images (70/105) carry neither a cryptographic artifact signature nor a provenance attestation, indicating the absence of any verifiable supply chain artifact.

C. Cross-Category Analysis (RQ2)

Table II reports mean scores aggregated by functional category and addresses RQ2. The most striking result is the gap between the GitOps/CD category — which achieves full build provenance (1.000), full artifact signing (1.000), and the highest source hygiene (0.829), for a composite mean of 0.910 — and the RAN & RIC category, which records the lowest composite mean in the corpus (0.139), with no signing and no provenance across any of its images. A particularly significant finding concerns the 5G/6G Core NF category, which implements the core session management, user authentication, and data routing functions of the network: its projects show zero build provenance (mean build = 0.000) and minimal artifact signing (mean artifact = 0.114), indicating that supply chain security controls have yet to be systematically adopted at this layer.

D. Per-Dimension Findings (RQ1)

a) F1 — Build provenance: Only 10.5% of images (11/105) carry a verifiable provenance attestation demonstrating adherence to the SLSA framework, which defines security requirements for the build process. Coverage is concentrated in three GitOps projects — Flux (6/6 components), Argo CD (1/1), and Argo Rollouts (1/1) — and in KubeEdge (3/4). Beyond these, no project in the corpus publishes verifiable SLSA provenance — including all Core NFs, RAN & RIC, NFV Orchestration, CN Networking, and zero-trust & PKI categories — placing the remainder of the 6G stack at build score 0.000, as reflected in the near-flat distributions in Fig. 2(b).

b) F2 — Artifact signing: Thirty-two images (30.5%) carry a cryptographic signature enabling verification of artifact authenticity and integrity prior to deployment. Signing is concentrated in the GitOps category (8/8 images, mean 1.000) and in the zero-trust & PKI category (16/18 images, mean 0.889). Projects such as OpenAirInterface and free5GC, which implement core 5G network functions, publish no signed images among those evaluated, a pattern visible in the largely bimodal distributions of Fig. 2(d).

c) F3 — Dependency exposure: This is the strongest dimension in the corpus, with a mean score of 0.791, reflecting generally sound dependency hygiene. However, a cluster of high-exposure outliers is identified — visible as individual points in Fig. 2(c): ONOS SD-RAN components score between 0.08 and 0.16, KubeEdge's edgemesh-agent scores 0.16, ONAP SO components score between 0.13 and 0.17, and both UERANSIM and oai-gnb score 0.29, indicating the presence of critical, actively exploited vulnerabilities in their dependency trees.

d) F4 — Source hygiene: The mean source score of 0.603 indicates moderate adoption of secure development practices and source repository hardening across the corpus, though with substantial within-category spread, as shown in Fig. 2(a). The GitOps category leads with a mean of 0.829; SD-Core achieves the highest individual project score at approximately 0.86. At the lower end, srsRAN scores 0.20 and OpenAirInterface approximately 0.37.

e) F5 — Composite score: The composite scores span a range of 0.807, from 0.114 (ONOS SD-RAN) to 0.921 (Flux), reflecting highly uneven supply chain maturity across the corpus. The GitOps/CD tier establishes the upper bound that the rest of the 6G stack has yet to reach.

E. Recurring Adoption Patterns (RQ1 → RQ3)

Beyond the per-dimension adoption figures reported across categories and projects, a significant finding concerns the shape of this adoption. Projects rarely implement the full set of controls covering the entire software development lifecycle; instead, adoption is partial and varies with project maturity or external drivers — regulatory compliance requirements, third-party security audits, or the emergence of relevant supply chain attacks — all of which may incentivize projects to invest in improving their supply chain posture. Against this backdrop,

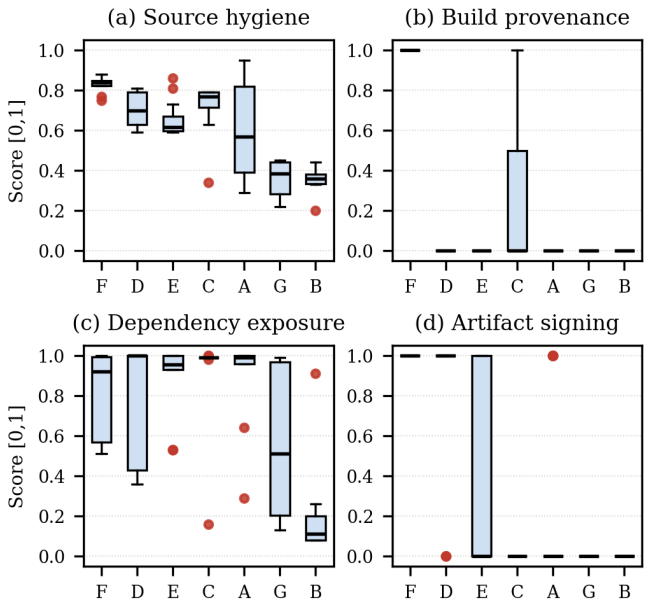


Fig. 2. Distribution of dimension scores by functional category (letters correspond to Table II). Box: interquartile range (IQR); line: median; whiskers: 1.5×IQR; red dots: outliers.

four distinct patterns of supply chain control adoption are identified. The patterns are not mutually exclusive: a single project may instantiate more than one (e.g. KubeEdge instantiates P2 and P3 simultaneously; Nephio and SPIFFE–SPIRE instantiate P1 and P3).

a) P1. Signing without provenance attestation: The majority of images that provide any supply chain verification control adopt artifact signing, while few attach a build provenance attestation. Twenty-six images across seven projects return a positive artifact score while their build score is 0.0 — Istio (5/5 components) and Cilium (4/5) being the clearest examples. The pattern is concentrated in the zero-trust & PKI and CN networking categories. Artifact signing introduces low friction into the software publication cycle: it requires only a signing call — such as cosign — to be added to the publishing pipeline. The practice of establishing 'who' produced an image appears to prevail over the practice of proving 'how' the image was built.

b) P2. Provenance or SBOM without signing: The inverse configuration also occurs. Some images provide build provenance attestations yet publish the image in the registry without a digital signature. KubeEdge achieves SLSA Level 3 and attaches a provenance attestation to three of its four analyzed components, yet none carry a digital signature. The project has made an effort to attach supply chain metadata to its images; however, verifying their authenticity and integrity would be more straightforward if a digital signature were also attached alongside the metadata in the registry — as a signed artifact enables a consumer to perform a single, lightweight verification step rather than the heavier path of retrieving, parsing, and validating the attestation independently.

c) P3. Inconsistent intra-project coverage: Some projects adopt signing or provenance for the artifacts they publish, but this adoption is not consistent across all components of the project. Four projects show a majority of signed components with at least one unsigned — Nephio (4/5; porch-server unsigned) being a representative case. KubeEdge exhibits good supply chain hygiene across all but one component: edgemesh-agent ships an older release, carries no provenance, and presents the highest critical vulnerability exposure in the project — substantially below the dependency scores of the other KubeEdge images. This inconsistency means that, from a deployment standpoint, it is not possible to assert that "project X signs all its images," which requires treating exceptions case by case and complicates the establishment of consistent admission policy gates in 6G environments.

d) P4. Repository hygiene not always aligned with artifact controls.: Good source repository hygiene — or sound development practices — does not always imply that other supply chain controls, such as image signing or provenance attestation, have been adopted. SD-Core is the clearest illustration: its ten components score between 0.82 and 0.95 on the source dimension yet return zero on both build and artifact. Computing the Pearson correlation across the corpus between source score and build score yields $r \approx 0.32$, and between source score and artifact score $r \approx 0.36$ — both weak. This finding justifies STELA-6G's design of evaluating the four supply chain dimensions independently, enabling a more precise and complete picture of the degree of care that projects apply to the images they publish.

e) Bridge to recommendations (RQ3): Examining all four identified patterns, no significant technical barrier prevents projects from implementing supply chain security best practices across all four supply chain dimensions. Yet adoption is selective and uneven. The GitOps category serves as the reference tier, demonstrating the path that other projects can follow to achieve a higher level of supply chain security maturity. Each identified pattern maps directly to a stakeholder group and a corresponding set of recommendations in Section VI: P1 and P2 inform recommendations for project maintainers (§VI-B); P3 and P4 inform admission policy-gate design for 6G platform operators (§VI-C).

VI. Discussion

A. Supply-Chain Posture: A Practical Assessment

The supply chain security posture of the evaluated corpus can be characterized as moderate overall, but with a pronounced stratification that prevents this aggregate figure from being interpreted uniformly. As shown in Sections V-B and V-C, adoption is neither uniform nor consistent — in several projects no coherent supply chain security strategy is apparent. Maturity is stratified by functional layer: the closer a project sits to the cloud-native delivery substrate (GitOps), the stronger its posture; the closer it sits to the RAN and Core NF layers, the weaker. Translated to a concrete deployability judgment, under a reasonable multi-dimensional

admission threshold — one that grants deployment only to images with a demonstrably trustworthy supply chain posture — GitOps and zero-trust PKI projects would pass today, while the majority of Core NF and RAN/RIC projects would not.

B. Recommendations for Project Maintainers

a) R1 — Adopt end-to-end pipelines combining signing and provenance.: As illustrated by pattern P1 (signing without provenance), several projects in the corpus sign their artifacts without attaching a build provenance attestation — Istio being one example. Signing is a valid and necessary control: it demonstrates that the image was published by the designated entity. What signing alone does not provide is visibility into how the image was built — the toolchain used, the build environment isolation, or the level of build integrity attained. Adding a cryptographically signed SLSA provenance attestation increases confidence that the artifact was built in a sound environment and reduces the likelihood of malicious tampering. Combining both controls is therefore the more practical choice, as it provides verifiable, easy-to-check metadata for downstream admission decisions: tools such as slsa-github-generator — used in the corpus by Flux, Argo CD, and KubeEdge — integrate seamlessly into CI/CD workflows and generate a SLSA Level 3 provenance attestation, while a cosign sign step completes the pipeline at minimal additional cost, consistent with broader industry initiatives such as Trusted Publishers [19], adopted by PyPI, npm, and RubyGems.

b) R2 — Ensure uniform, self-sustained supply chain coverage.: Pattern P3 demonstrates that a single unsigned component breaks the assurance that "project X signs all its images" — as seen, for example, in Nephio, where all analyzed images carried a signature except porch-server. All components intended for publication should therefore be included in the signing pipeline in a consistent and uniform manner, including less actively maintained components, which are precisely the ones most likely to fall outside an existing pipeline. Embedding this hardening — provenance attestation, artifact signing, SBOM generation — into a project's own release criteria, rather than treating it as an external, one-off compliance effort, is a more sustainable approach that ensures security practices are maintained as the project evolves.

C. Recommendations for 6G Platform Operators

a) R4 — Implement multi-dimensional admission policy gates.: Pattern P4 shows that a high source score does not predict strong signing or provenance practices — EdgeX Foundry being a clear example. Evaluating only a single dimension of the supply chain produces an incomplete picture, because relying on any one score as a proxy for overall security can lead to classifying as trustworthy an artifact that is not. This is particularly critical in 5G/6G environments where zero-trust principles are applied: platform operators should therefore enforce policy gates that check all four supply chain dimensions independently – scores STELA-6G reports individually alongside the composite – yielding a complete picture of an image's posture prior to admission.

b) R5 — Define tiered admission policies by component criticality.: Not all components require the same level of supply chain security. Stratified policies can be defined in which the most critical network functions are subject to stricter requirements — for instance, mandatory image signing combined with at least SLSA Level 2 build provenance. Demanding SLSA Level 3 across all components may not be realistic today; a more practical approach is to establish differentiated thresholds — the highest for GitOps and PKI components, which already meet this bar, and a minimum transitional level for Core NF and RAN components, with mandatory signing as the first required step and a defined improvement roadmap. The per-category baselines shown in Table II provide a concrete starting point for calibrating these thresholds.

D. Sensitivity to Scoring Weights

As described in Section IV-C, the composite score was computed using equal weights across the four dimensions. Alternative weighting schemes are also plausible — for instance, prioritizing the build dimension, since malicious code injected into a compiled artifact can have catastrophic consequences, or the artifact dimension, since signature verification often serves as the final admission gate. Perturbing the equal-weight baseline within a balanced range (adjusting one weight by up to ± 0.1, compensated across the others) does not alter the study's central conclusions, confirming that the observed functional gap and adoption patterns are properties of the corpus rather than artifacts of the chosen weighting scheme.

E. Open Challenges for the Research and Standards Community

a) C1— Mapping technical supply chain practices to regulatory frameworks.: The European Cyber Resilience Act establishes transparency and security requirements for the software supply chain of products with digital elements; however, translating these regulatory requirements into specific, verifiable technical artifacts remains an open research problem. It is not yet established, for example, what minimum SLSA level or what combination of STELA-6G dimension scores constitutes sufficient evidence of compliance for a given CRA obligation. Bridging this gap between technical measurement and legal demonstrability is a necessary step for operationalizing supply chain regulations in 6G deployments.

b) C2— Trust models for third-party re-publishing.: Some relevant 5G/6G projects publish their source code but do not publish their own container images, delegating this to third parties. Current supply chain frameworks — including STELA-6G — assume that the image publisher is the software developer; when this assumption breaks down, no established framework provides a model for evaluating the chain of custody. This is an open research challenge, and developing trust models for third-party re-publishing scenarios is relevant for the 6G ecosystem, where operators and system integrators

frequently redistribute images originally produced by open-source communities.

F. Limitations of the Study

a) T1 — External validity.: The study evaluates a curated selection of prominent open-source projects, assembled as a feasible, representative approximation of the major functional layers of the 6G cloud-native stack. Closed-source network function images, moreover, do not permit independent external evaluation of their SDLC chain. Per-category sample sizes are also small and uneven (ranging from 8 to 35 images across categories), limiting the precision of cross-category comparisons. The results should therefore be interpreted as indicative of supply chain practices across the OSS 6G ecosystem, rather than a definitive characterization of it.

b) T2 — OSM source score as a best-effort approximation.: The OSM source repository is hosted on a private GitLab instance inaccessible to the OpenSSF Scorecard online service; the score was obtained using scorecard –local, which omits several checks available only in online mode. As a result, this indicator is not directly comparable to those of other projects.

c) T3 — OpenSSF Scorecard run variability.: Source scores may fluctuate slightly between evaluation runs, as several Scorecard checks assess recent repository activity in actively maintained, evolving projects.

VII. Conclusions

This paper presents an empirical assessment of the supply chain security posture of the open-source 6G ecosystem across four dimensions of the software supply chain — source hygiene, build provenance, dependency exposure, and artifact signing. The verdict is that security maturity is stratified by functional layer: strongest closest to the cloud-native delivery substrate (GitOps), and progressively weaker toward the RAN and Core NF layers. One of the study's most striking figures is that 66.7% of the evaluated images carry neither a cryptographic signature nor a provenance attestation — the two supply chain artifacts most directly verifiable at deployment time. A tool such as STELA-6G can effectively measure supply chain security across these four dimensions and serve as an actionable instrument for integration into 6G deployment pipelines.

As future work, we identify the integration of STELA-6G as a policy gate within CI/CD pipelines and network service orchestration admission control: the deployment of a network service would be conditioned on the posture score of the container images supporting it, enacting a zero-trust security model for the deployment of 6G network services. Finally, extending the corpus with additional projects — for instance, O-RAN SC[1], once its registry becomes reachable — and conducting a longitudinal, per-release evaluation would allow the evolution of the ecosystem's supply chain security to be tracked over time as the 6G software ecosystem matures until commercial operation.

[1]https://o-ran-sc.org/

Acknowledgment

This work is supported by the Smart Networks and Services Joint Undertaking (SNS JU) under the European Union's Horizon Europe research and innovation programme under Grant Agreement No 101139198, iTrust6G project. This work is also supported by Departament de Recerca i Universitats, Generalitat de Catalunya. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union, SNS-JU or Generalitat de Catalunya. Neither the European Union, Generalitat de Catalunya nor the granting authority can be held responsible for them.

References

[1] S. Plastras, P. Pavlidis, *et al.*, "An Open-Source CAMARA UE Location API Implementation Over 3GPP NEF for AI-Native Crowd Assessment in 6G Networks," in *2026 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, pp. 588–593, June 2026. ISSN: 2575-4912.

[2] S. Hendrick and R. Haiby, "The 2025 Open Source Networking Study: The Role and Value of Open Source in the Networking Industry's Software Stack," tech. rep., The Linux Foundation, Apr. 2025.

[3] "NIS 2 Directive." https://www.nis-2-directive.com/.

[4] "Cyber Resilience Act | Shaping Europe's digital future," Sept. 2022. https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act.

[5] D. Company Estall and M. Compastié, "STELA-6G — Supply Chain Corpus & Analysis Artifacts," Aug. 2026. https://zenodo.org/records/22112421.

[6] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "SoK: Taxonomy of Attacks on Open-Source Software Supply Chains," in *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 1509–1526, May 2023.

[7] L. Williams, G. Benedetti, *et al.*, "Research Directions in Software Supply Chain Security," *ACM Transactions on Software Engineering and Methodology*, vol. 34, pp. 1–38, June 2025.

[8] Cloud Native Computing Foundation Tag Security, "The Secure Software Factory: A Reference Architecture to Securing the Software Supply Chain," Mar. 2023.

[9] European Union Agency for Cybersecurity., *Good practices for supply chain cybersecurity.* LU: Publications Office, 2023.

[10] M. Souppaya, K. Scarfone, and D. Dodson, "Secure Software Development Framework (SSDF) version 1.1 :: recommendations for mitigating the risk of software vulnerabilities," Tech. Rep. NIST SP 800-218, National Institute of Standards and Technology (U.S.), Gaithersburg, MD, Feb. 2022.

[11] N. Zahan, T. Zimmermann, *et al.*, "What are weak links in the npm supply chain?," in *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, (Pittsburgh Pennsylvania), pp. 331–340, ACM, May 2022.

[12] K. G. Kalu, T. Singla, *et al.*, "An Industry Interview Study of Software Signing for Supply Chain Security," pp. 81–100, 2025.

[13] A. Mills, J. White, and P. Legg, "Longitudinal risk-based security assessment of docker software container images," *Computers & Security*, vol. 135, p. 103478, Dec. 2023.

[14] N. Zahan, P. Kanakiya, *et al.*, "OpenSSF Scorecard: On the Path Toward Ecosystem-Wide Automated Security Metrics," *IEEE Security & Privacy*, vol. 21, pp. 76–88, Nov. 2023.

[15] W. Enck and L. Williams, "Top Five Challenges in Software Supply Chain Security: Observations From 30 Industry and Government Organizations," *IEEE Security & Privacy*, vol. 20, pp. 96–100, Mar. 2022.

[16] K. Ramezanpour and J. Jagannath, "Intelligent zero trust architecture for 5G/6G networks," *Computer Networks*, vol. 217, p. 109358, Nov. 2022.

[17] J. Reed, P. Scalise, *et al.*, "A Systematic Survey on 5G and 6G Security Considerations, Challenges, Trends, and Research Areas," *Future Internet*, vol. 16, p. 67, Feb. 2024.

[18] S. Klados, L. Ferro, *et al.*, "D4.1 - Primary Report on Intelligent Trust Management Framework," July 2025. https://www.sns-itrust6g.com/wp-content/uploads/2025/11/iTrust6G-D4.1.pdf.

[19] Python Software Foundation, "Trusted Publishers," 2024. https://docs.pypi.org/trusted-publishers/.