ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO4BC003

PROTECTING YOUR CONTROLS INFRASTRUCTURE SUPPLY CHAIN

B. Copy *, J.B. de Martel, F. Ehm, P. Elson, S. Page †, M. Pratoussy, L. Van Mol
CERN Beams Department, 1211 Geneva, Switzerland

Abstract

Supply chain attacks have been constantly increasing since being first documented in 2013. Profitable and relatively simple to put in place for a potential attacker, they compromise organizations at the core of their operation. The number of high profile supply chain attacks has more than quadrupled in the last four years and the trend is expected to continue unless countermeasures are widely adopted.

In the context of open science, the overwhelming reliance of scientific software development on open-source code, as well as the multiplicity of software technologies employed in large scale deployments make it increasingly difficult for asset owners to assess vulnerabilities threatening their activities.

Recently introduced regulations by both the US government (White House executive order EO14028) and the EU commission (E.U. Cyber Resilience Act) mandate that both Service and Equipment suppliers of government contracts provide Software Bills of Materials (SBOM) of their commercial products in a standard and open data format. Such SBOM documents can then be used to automate the discovery, and assess the impact of, known or future vulnerabilities and how to best mitigate them.

This paper will explain how CERN investigated the implementation of SBOM management in the context of its accelerator controls infrastructure, which solutions are available on the market today, and how they can be used to gradually enforce secure dependency lifecycle policies for the developer community.

INTRODUCTION

Supply chain attacks involve a malicious third-party infiltrating an organization by exploiting vulnerabilities in third-party components or software used in critical systems. These attacks brought in the context of accelerator controls can compromise the integrity and reliability of operations, potentially leading to disruptions, data breaches, and unauthorized control over essential infrastructure.

The first documented software supply chain attack in June 2013 caused a distributed denial of service on the South Korean government and several news outlets [1] by leveraging the legitimate auto-update mechanism of the *SimDisk* file-sharing and storage service. When organizations rely on third-party suppliers for components, software, or services, they trust that these suppliers have adequate security measures in place. Suppliers with inadequate practices become vulnerable points of entry for attackers.

In the context of open science and its major reliance on open-source software, open-source software maintainers replace commercial suppliers as essential pieces in securing the supply chain. While commercial suppliers are akin to regulated factories producing reliable components, open-source maintainers act as skilled artisans crafting components collaboratively. Their level of dedication and engagement is not dictated by commercial agreements and can vary over time, yet open source software constitutes the cornerstone of CERN accelerator operations.

Furthermore, this reliance on open-source software is globally following an accelerating trend, without signs of stabilising. The 2023 Open Source Security and Risk Analysis (OSSRA) on open source trends [2] reports that in 2022, the average software project depends on 595 third-party open-source libraries (a near 200% increase over the last four years), while 48% of the analyzed projects contained high-risk vulnerabilities such as documented proof-of-concept, active exploits or remote code execution opportunities (a minor decrease of 2% since 2021). The OSSRA 2023 report also shows that :

- Organizations are insufficiently fixing high-risk vulnerabilities, with a global 42% increase of their presence in codebases since 2018. Some industries more lucrative to attackers, such as e-commerce, are even reporting a 557% increase over the last five years.
- Open source maintenance is on the decline, with an increasing usage of open-source components that have been without activity over the past 24 months, from 85% in 2018 to 90% in 2022. Reasons most cited by open source maintainers are a lack of recognition and inadequate compensation.

In addition to the inexorable discovery and exploitation of software defects leading to vulnerabilities, open source software is subject to neglect with dreadful consequences. Lack of open source software maintenance opens the door to abuse such as the introduction of malware, *protestware* (software that contains some kind of political protest in addition to its primary function), *typo-squatting* (software that relies on common typographic or spelling errors to spread) and dependency confusion (when a malicious dependency library is downloaded from a public registry rather than the intended private/internal registry, for instance by exhibiting a higher version number).

Without appropriate countermeasures, such abuse is :

- Hard to detect and report upon, as both a suitable dependency taxonomy and communication channels need to be established.
- Difficult to track and to mitigate without the usage of an inventory and automation.

* brice.copy@cern.ch
† stephen.page@cern.ch

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO4BC003

- Taken less seriously by the owners of isolated controlled infrastructure, as their cultural take on software vulnerabilities does not account for its destructive intra-muros capabilities.

Standards and measures that have proven effective in the industry to address supply chain concerns will be described in the following sections, including how they can be applied in a research-focused, accelerator controls context, such as CERN.

SOFTWARE BILL OF MATERIALS

Nowadays, using dependency management as part of a build system is an integral part of software development. This automates the retrieval and integration of external libraries, ensuring consistent and up-to-date dependencies. In turn, this greatly simplifies the tracking of software components and therefore, also facilitates the means to patch and upgrade their application.

Nevertheless, build systems are programming language dependent and are therefore sub-optimal when applied to a language which they were not designed for, resulting in inefficient management of that language's third-party dependencies. However, modern software systems routinely combine multiple technologies to accomplish their objectives e.g. Java for enterprise back-end systems, Python for data science, Typescript for web-based user interfaces, etc.

A Software Bill Of Material (SBOM) is a machine-readable document that provides a unified, platform-agnostic view of software dependencies. SBOM descriptors bring the following advantages [3, Chapter 4] :

- Comprehensive Visibility: SBOMs provide a holistic view of all software components and dependencies across different platforms and technologies. This comprehensive visibility is crucial for identifying vulnerabilities that might exist in various parts of the system.

- Risk Mitigation: By proactively identifying and addressing vulnerabilities across the entire software stack, the risk of security breaches, data leaks, and operational disruptions can be significantly reduced. Identifying and addressing the vulnerabilities in a platform-agnostic manner can reduce the attack surface of the overall system, making it harder for attackers to find and exploit weaknesses.

- Efficient Patch Management: An SBOM helps to efficiently track and manage patches or updates for vulnerable components. Remediation of vulnerabilities can be prioritized based on their criticality, regardless of the platform, reducing the risk of exploitation.

- Future-Proofing: As technology evolves, organizations often adopt new platforms or languages. A technology agnostic SBOM is adaptable and future-proof, thus it can help ensure that security practices are scaled and adapted to technological changes.

- Compliance and Reporting: Many regulatory frameworks and security standards require organizations to maintain a comprehensive inventory of software components and their security status. A technology agnostic SBOM simplifies compliance efforts by providing a centralized source of truth.

Having an accurate SBOM facilitates the discover of existing vulnerabilities within a software system. In turn, it can then be used to cross-check the existing production software system against constantly emerging vulnerabilities. While Zero-day vulnerabilities certainly grab headlines, the so-called N-days vulnerabilities are much more prevalent and tend to linger for much longer, just as the active interest in seeing them eliminated tends to wane with time. This is all the more concerning in accelerator controls facilities. [4]

The potential impact of these improvements has led to multiple national and transnational government bodies to adopt it in their legislations. In May 2021, the US government issued Executive Order 14028, "Improving the Nation's Cybersecurity", mandating the publication and usage of SBOMs amongst federal agencies, federal contractors and critical infrastructure. In September 2022, the European Union has in-turn, adopted the Cyber Resilience act that makes SBOM an integral part of the CE product certification process [5].

A Bill Of Material (BOM) is not a new concept : it is already used in a variety of industries, such as manufacturing. Software never operates in isolation, it relies on underlying hardware, a supporting regulatory framework (*e.g.* service level agreements, value chain portfolios, chains of trust etc.), as well as third-party services. All these concepts and more, can also be captured as part of a larger bill of material. Certain BOM standards, such as CycloneDX [6], provision specifically for such wider contexts to support a complete secure life cycle.

In the context of the CERN accelerator controls software, since 2022, a small portfolio of significant applications has been selected for analysis, to evaluate the steps required for SBOM generation and in-turn, assess the usefulness of the resulting risk assessment.

ANALYSIS OF SAMPLE CERN ACCELERATOR CONTROLS SOFTWARE

Software Project Portfolio

In order to determine the feasibility of consistently producing SBOMs and how much insight their analysis would bring to accelerator operation, a broad selection of applications have been enhanced with the necessary build steps to produce and upload SBOMs to a **Dependency Track** [7] instance. **Dependency Track** is an open-source analysis platform that allows organizations to identify and reduce risks in the software supply chain. It ingests SBOM documents and continuously monitors their contents for known and emerging vulnerabilities, integrating with both public and private vulnerability databases.

The portfolio of CERN accelerator controls software

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO4BC003

projects selected for review and analysis in Dependency Track included :

- **Project A** : a modular Java framework for monitoring large-scale infrastructure.
- **Project B** : A messaging and alarm propagation service for the CERN accelerator complex.
- **Project C** : A list of recommended Java software libraries that developers are advised to integrate in their systems, to ensure maximum software compatibility for the duration of a given LHC run.
- **Project D** : A service to manage operational settings for the CERN accelerator complex.
- **Project E** : A Python-based machine learning infrastructure, allowing the deployment of custom payloads (TensorFlow, PyTorch, *etc...*).
- **Project F** : A Docker container base image, recommended for CERN accelerator control applications, based on CentOS Stream 8. It is part of pilot portfolio as a baseline for container scanning, which is not used in operation to date.

Key Metrics

Several metrics are available as part of the Dependency Track reports, however for the purposes of this paper, two key metrics have been selected for consideration :

1. Percentage of vulnerable components : how many components (*i.e.* software libraries the application depends on) are subject to known vulnerabilities. This includes trivial and non exploitable vulnerabilities. This metric illustrates the fact that certain key components tend to be difficult to implement with unequivocal correctness and therefore attract the attention of malicious actors and security researchers.
2. Maximum Common Vulnerability Score (CVSS) : the Common Vulnerability Scoring System (CVSS) represents in absolute terms, how easily a given vulnerability can be exploited. It is expressed on scale of 0 to 10, with 10 representing a target system that can be totally compromised (*i.e.* grant the attacker full control) with no other pre-requisite than being able to communicate with the system in question (*i.e.* opening a network socket on the advertised service port).

A summary of the range of results from the vulnerability scanning is shown in Table 1.

Table 1: Summary of Portfolio Risk Indicators

Vulnerable Components	Max CVSS
1% to 22%	5.5 to 10

Interpretation and Mitigations

As illustrated in Table 1, some of the selected projects exhibit **significant and immediate risk**, the only mitigating factor being that they are being operated on a segregated operational network. Access control, and cybersecurity in general, on the CERN operational network is considered as a safeguard against operational mistakes rather than a protection against malicious intent.

The most critical vulnerabilities in this selected portfolio come from components that :

- could be upgraded to a later, safer version **in every single case**; whether the upgrade is technically feasible or not requires significant expertise and in some cases may not be possible during operational running periods of the accelerator complex.
- exhibit security-related functionality, typically using security frameworks that handle authentication and authorization filters (*e.g.* Spring security).
- exhibit low-level data functions, such as serialization libraries (*e.g.* *Snakeyaml*, *Xstream*) or logging (*e.g.* *log4j*).
- require active involvement from a malicious party (which in the context of CERN reduces significantly the overall risk).

A specific note, which will be referred to later, should be made of the scanning of the **Project F** container image, which revealed a surprisingly low number of known vulnerabilities, considering the relative obsolescence of the given operating system version. Upon manual inspection, the image did carry critically vulnerable libraries (*e.g.* *openssl v1.1.1*) that were not correctly identified and reported upon. Using another scanning tool (Syft [8]) revealed a much larger number of vulnerabilities, but unfortunately did not report the one identified by Trivy [9].

With dependency inventories and metrics readily available, consideration now needs to be given to how to act efficiently upon identified risks and how to instill accountability across the board.

SHIFT LEFT CULTURE

In an agile environment such as CERN, developers aim to be empowered and are expected to take an active role not only in the development and deployment of releases, but in ensuring the security of the software they develop. Infrastructure administrators are no longer necessary, nor mandated to take part in deployments. This reflects the guiding DevOps principles of automation, collaboration, and shared ownership, ultimately leading to more secure and resilient software products. This redistribution of responsibilities is referred to as **shift left culture** [10, Section 1.4] and permeates through all levels of the software development life cycle:

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO4BC003

- **Continuous Integration and Continuous Delivery** (CI/CD) : In DevOps, the goal is to automate and accelerate the software development and delivery pipeline. "Shift left" concepts align perfectly with this by integrating security into the CI/CD process, ensuring that security checks are performed automatically as code is developed, tested, and deployed.

- **Collaboration** : DevOps promotes collaboration between development, operations, and security teams. "Shift left" encourages early and continuous collaboration by involving security experts in the development process, fostering a culture of shared responsibility for security.

- **Agile iterative development** : In methodologies employed at CERN, such as Scrum and Kanban, software is developed and updated at a rapid pace. A "Shift left" approach helps identify and remediate security issues early, allowing teams to iterate quickly without compromising security. Integration with code review facilities employed at CERN, such as Gitlab merge requests, is a means to ensure that vulnerability scanning becomes an integral part of validating new code contributions and curtail the deployment of known vulnerabilities.

- **Auditing and Compliance** : Shift left practices generate audit trails and records that can be valuable for compliance and regulatory purposes, ensuring that security is not an afterthought when it comes to meeting industry standards and requirements.

- **Cost-Efficiency** : By catching security issues early in the development and deployment process, "Shift left" helps reduce the cost of fixing vulnerabilities and security-related problems at later stages, which can be significantly more expensive.

Adopting **Shift left culture** in combination with SBOM not only enhances security but also contributes to the overall agility, efficiency, and quality of software development and infrastructure management processes. There are however, numerous obstacles to attaining these goals in academic research environments, such as CERN.

STEPS TO ADOPTION

Initial investigation into vulnerability scanning and SBOM usage at CERN indicate clear benefits towards better risk awareness, accountability and transparency. Nevertheless, a number of factors still need to be addressed for this practice to become wide-spread.

Inventory and Ownership

Clear ownership is a foundational element of effective software security. It establishes accountability, responsibility, and visibility, enabling organizations to proactively manage security risks, respond to incidents, and maintain a strong security posture throughout the software development life cycle. While service ownership at CERN is most often clearly specified at a management level, this notion becomes quickly diluted when working down to an individual software component, without a formal centralized definition of how the component contributes to which service as a whole. This lack of fine-grained information results in a poor visibility and accountability for component owners, which in turn deprives the organization of compliance enforcement and resource-aware risk management. An SBOM standard such as **CycloneDX** does provision for such information, but it comes in orthogonally with respect to wider enterprise management frameworks (*e.g.* ITIL, TOGAF) without clear integration to date.

Standards Compliance And BOM Accuracy

At the time of writing, three major SBOM standards are competing for widespread adoption : CycloneDX, SPDX and SWID. Certain tools support only a given standard and rely on third-party conversion mechanisms. SBOM generation for a single software product by different tools can also result in completely different component lists, with missing elements. SBOM documents produced by two different container image scanning tools (Trivy [9] and Syft [8]) after scanning the very same container image can differ significantly. The resulting vulnerability reports are therefore different, with some parts even featuring contradicting or non-standard information. As illustrated in the case of the container image **Project F** in subsection "Interpretation and mitigations", identifying components accurately when their fingerprint, name and expected locations can be particularly difficult, and may require employing a series of heuristics [11]. This further stresses the need to adopt "Shift left" strategies, as the original software authors and maintainers are best placed to produce accurate SBOM documents and act upon resulting vulnerability reports.

Tooling and Integration

While SBOM standards are maturing, they remain relatively new and lack the appropriate tooling support if they are to thrive in large scale research environments.

Reporting SBOM can reveal a plethora of risk metrics, but unfortunately suffer from a lack of adequate reporting models and tools. Existing SBOM management platforms such as Dependency Track [7] are both opinionated and deficient, as they do not support multi-dimensional data aggregation, complex filtering or simply perform data exports, that in turn, could be exploited in rich reporting platforms (e.g. Tableau, BusinessObjects). This limits compliance reporting and vulnerability auditing, and therefore fails to exploit the possibilities offered by the CycloneDX standard, for instance.

Scalability Sensitive information management requires the highest level of granularity when it comes to accessing, asserting provenance, and following up existing and emerging risks. When compared to other cybersecurity risk

ISBN: 978-3-95450-238-7 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2023-MO4BC003

management solutions, SBOM solutions like Dependency Track [7] or Quay [12] are still recent and do not provide scalable access rights management, nor support information partitioning and role-based access according to an organization's existing legal and functional hierarchies. In contrast, established software repository solutions such as Artifactory are much more advanced, but unfortunately lack the support for SBOM management and vulnerability auditing.

Suitable Information Channels Once ownership and reporting of cyber risks are in place, the continued life cycle and evolving impact of a vulnerability must be audited. Platforms such as Dependency Track provide rudimentary and inflexible support, without any means to integrate custom business process logic. Most organizations already have risk management procedures, escalation processes and dissemination workflows, yet Dependency Track ignores them entirely, and leaves it up to the implementer to integrate them through its REST API or industry standard solutions (*i.e. webhooks, JIRA issue tracking etc.*).

CONCLUSION

This paper summarizes the work done at CERN to gain practical experience in better protecting accelerator controls software against supply chain vulnerabilities. The conducted investigations revealed the immediate benefits of using Software Bills of Materials and integrating a full vulnerability auditing cycle in the development process. At the same time, this paper described the shortcomings identified with the current state of the art. Looking ahead, these shortcomings must be addressed if such practices are to find their rightful place amongst other tools in the cybersecurity risk mitigation arsenal.

REFERENCES

[1] Trend Micro Investigates June 25 Cyber Attacks in South Korea, https://www.trendmicro.com/vinfo/us/threat-encyclopedia/web-attack/

[2] Open Source Security and Risk Analysis Report 2023, https://www.synopsys.com/software-integrity/engage/ossra/rep-ossra-2023-pdf

[3] C. Hughes *et al.*, *Software Transparency*. NJ, USA: Wiley, 2023.

[4] The Overlooked Problem of N-days Vulnerabilities, https://www.darkreading.com/vulnerabilities-threats/the-overlooked-problem-of-n-day-vulnerabilities

[5] Cyber Resilience Act, https://digital-strategy.ec.europa.eu/en/library/cyber-resilience-act

[6] CycloneDX Specification, https://cyclonedx.org/specification/overview/

[7] Dependency Track Project, https://owasp.org/www-project-dependency-track/

[8] Syft and Grype, Developer-friendly Scanning Tools For Container Image Security, https://anchore.com/opensource/

[9] Trivy The All-In-One Open Source Security Scanner, https://trivy.dev/

[10] D. Fisher, *Application Security Program Handbook*. NY state, USA: Manning Publications, 2023.

[11] How NetRise Uses Knowledge Graphs to Identify Components in SBOMs, https://www.netrise.io/

[12] Quay, a Scalable Open Source Platform to Host Container Images, https://www.projectquay.io/