SLSA++
A Survey of Software Supply Chain Security Practices and Beliefs

A Joint Survey by:

Chainguard ECLIPSE FOUNDATION Rust Foundation OpenSSF OPEN SOURCE SECURITY FOUNDATION

The Rust Foundation is dedicated to a more secure, efficient, and performant future through the Rust programming language. We take software supply chain security seriously, and reports such as this help us to shape our proactive approach as part of our ongoing Security Initiative. It is clear from this report that, while security is seen as a priority, there is still some way to go to ensure that it is pursued coherently and efficiently. Although Rust already holds great promise for a more secure global supply chain, it will remain essential to explore and implement best practices as Rust continues to grow in adoption, where SLSA may play a dominant role. The Rust Foundation looks forward to working with interested stakeholders on effective and impactful solutions for security.

Rebecca Rumbul
Executive Director & CEO
Rust Foundation

At the Eclipse Foundation, we believe that foundations have an important role to play in addressing the challenges of securing open source and its supply chain. Specifically, our focus as a foundation is to provide services to our projects that help improve their security posture based on our Open Source Software Supply Chain Best Practices; along with the implementation of a SLSA-based badging program for Eclipse Foundation projects.

However, this is just a starting point as software security is a never-ending process. We are pleased to have been a partner in fielding this survey which provides some interesting insights that will help us and other stakeholders to continue to improve processes and evolve best practices around open source supply chain security.

Mikael Barbero
Head of Security
Eclipse Foundation

When we drafted the initial proposal for SLSA, our goal was to improve the state of software supply chain security across the industry, particularly open source, and to defend against the most pressing integrity threats. With SLSA, organizations can adopt the framework for their own internal best practices and also make informed choices about the security posture of the software they consume.

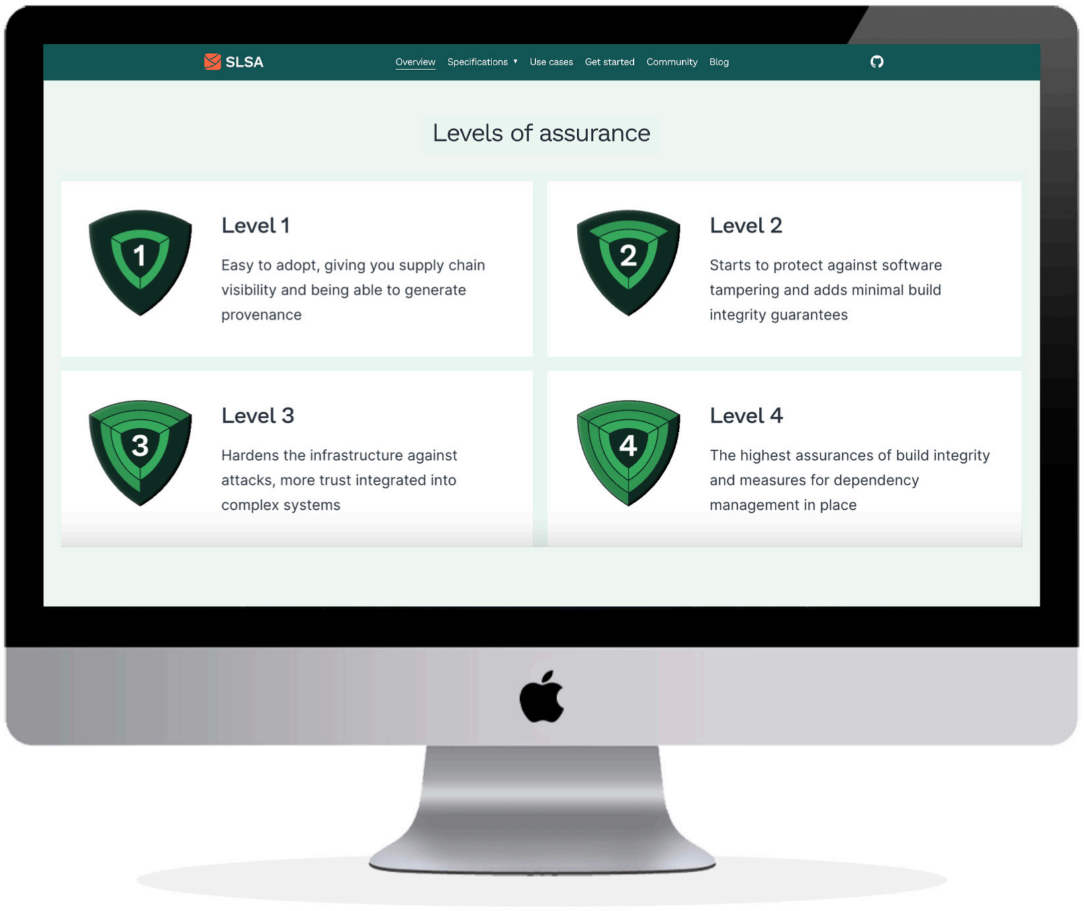
The results of the first SLSA usage survey show me that while SLSA is helpful there is still a lot of work that needs to be done to integrate common software supply chain security practices into the development lifecycle. We are definitely trending positively, but this report illuminates critical gaps and challenging areas that we can start addressing today to ensure a more secure software supply chain tomorrow

Kim Lewandowski
Co-Founder and Head of Product at Chainguard
& Co-Founder of SLSA

Executive Summary

Is everyone doing software supply chain security? Or is everyone just talking about software supply chain security? And do software professionals actually think different software supply chain practices are helpful? Easy or difficult?

To date, answering these questions systematically has been difficult. To find some answers, this survey asked 167 software professionals from around the world, at organizations of varying sizes and with various degrees of commitment to software supply chain security. All respondents answered a series of questions inspired by the Supply-chain Levels for Software Artifacts (SLSA) framework. These questions investigated the extent to which respondents used SLSA-related and other software supply chain security practices, the extent to which respondents perceive these practices as helpful, and the extent to which respondents believe these practices are difficult.



Levels of assurance

Level 1

Easy to adopt, giving you supply chain visibility and being able to generate provenance

Level 2

Starts to protect against software tampering and adds minimal build integrity guarantees

Level 3

Hardens the infrastructure against attacks, more trust integrated into complex systems

Level 4

The highest assurances of build integrity and measures for dependency management in place

Three findings stand out:

1. Some software supply chain security practices are widely practiced. Many SLSA security practices already enjoy at least moderate adoption. That said, providing provenance, arguably a key SLSA-related practice, notably lags in adoption.

2. Most SLSA practices are considered helpful though there is hardly any variation in the aggregate level of perceived usefulness. Importantly though, an individual’s belief in the usefulness of a SLSA practice is closely related to whether they or their organization adopt it.

3. Some SLSA practices are considered substantially more difficult than others. For instance, using a centralized build service or performing container vulnerability scanning are considered relatively easy while hermetic and reproducible builds and providing provenance are considered notably difficult. Surprisingly, the perceived difficulty of each SLSA practice has relatively little effect on whether an individual or organization adopts a practice. The results suggest that where there is a will, there is a way.

In short, there are some software organizations actually adopting software supply chain security practices, most SLSA practices are perceived as useful, and some practices are considered quite difficult while others are considered easy.

The report below performs a broader introduction and motivation of the topic, explains the methodology, describes the population surveyed, and then summarizes the analytical findings.

Introduction

Software supply chain security has achieved meme-like status over the past couple years. Those who work in the software industry now have LinkedIn feeds featuring sprawling comment threads about, for instance, software bill of materials (SBOMs) or open source software vulnerabilities. This ferment is exciting and should be welcomed. Now that most organizations and individuals have built businesses and lives with crucial dependencies on the Internet, computing, and software, it's only reasonable that these same organizations and individuals are asking for safety, including from compromises of the software supply chain.

But as with all topics that achieve the status of meme, it can be hard to separate the reality of now from the potential of tomorrow and, consequently, difficult to answer even basic questions. For instance, is software supply chain security merely a distant hope, a set of approaches that is preached but not practiced? How difficult do software professionals find it to implement software supply chain security practices? And do software professionals even find these practices helpful?

In the hope that answers to these questions will enrich the already bubbling dialogue on this topic, this survey sought to shed light on the here and now of software supply chain security. Toward that end, the survey adopted the Supply-chain Levels of Software Artifacts (SLSA, pronounced "salsa") framework as the organizing idea underpinning the survey. Housed within the Open Source Security Foundation (OpenSSF), SLSA is a framework for software supply chain integrity, ensuring that, in the language of the project, "the source code you're relying on is the code you're actually using." SLSA, in its v0.1 form, describes a series of "requirements" for achieving ever higher levels of protection. The survey therefore focused on these requirements, asking participants to report on their use and attitudes towards these requirements. Because SLSA does not include all aspects of software supply chain security, the survey also added some questions, hence SLSA++, about other popular practices sometimes associated with software supply chain security, including SBOMs and vulnerability scanning.

Finally, this survey is neither the first nor the last on the topic of software supply chain security. Previous academic research has shed light on the perceived utility and cost of a wide range of open source software security practices and industry research has highlighted the immaturity of current open source software security practices. The 2022 Google/DORA State of DevOps report, released after we had conducted this survey, even focused on SLSA, though not specifically on attitudes towards the difficulties and benefits of SLSA.

The next sections of the report describe the methodology, the survey sample, and the key findings.

Methodology

The survey questionnaire focused on identifying and understanding the characteristics of the respondents and then understanding how frequently the respondents reported using the different SLSA requirements and whether the respondents perceived these practices as difficult or easy and helpful or not. Table 1 identifies the different security requirements in the survey and the definition presented to the survey respondents.

TABLE 1.
SLSA++ Requirements in the Survey and Associated Definition Presented to Survey Respondents

Requirement Name	Definition Presented to Survey Respondents
Two person review	Every change in the revision history must be agreed upon by two trusted persons prior to submission. The contributor counts towards the total if the contributor is trusted.
Build service	All build steps must be run on a build service and not on a developer's workstation. CircleCI is an example of a build service.
Ephemeral	All build service steps must be run in an ephemeral environment such as a container or virtual machine provisioned solely for this build and not reused from a prior build. Most common continuous integration systems (e.g. GitHub Actions) are compliant.
Isolated	Build steps must be run in an isolated environment free of influence from other build instances.
Hermetic	All transitive build steps, sources, and dependencies must be fully declared with immutable references and the build steps run with no network access. This is sometimes called a "hermetic" build.
Reproducible	Re-running the build steps with identical input artifacts must result in bit-for-bit identical output. This is called a reproducible build.
Provenance available	The provenance information must be available to the consumer. Provenance is verifiable information about software artifacts describing when, where, and how something was produced. More information can be found here: https://slsa.dev/provenance/v0.2
Signatures	You must sign artifacts you build. *This requirement is not currently part of SLSA.*
Vulnerability scanning	All container images must be scanned for vulnerabilities. *This requirement is not currently part of SLSA.*
SBOM	All artifacts built by your organization must have a corresponding Software Bill of Materials (SBOM), an "ingredient list" of the components in the artifact. *This requirement is not currently part of SLSA.*

For each security requirement, the respondent answered three questions:

1

How often do you use this practice?
(On a scale of 1-5. 1=never, 5=always)

2

In your opinion, how helpful is this practice for preventing, mitigating and/or remediating software supply chain security attacks?
(On a scale of 1-5. 1=not helpful at all, 5=Extremely helpful)

3

In your opinion, how difficult is it to implement this practice?
(On a scale of 1-5. 1=not difficult at all, 5=Extremely difficult)

The online survey was distributed widely by the organizations that sponsored this survey including via email and social media during the summer and fall of 2022. The survey specifically noted that it was “intended for professionals involved in the creation and maintenance of software, whether directly or indirectly.” The survey also asked respondents to answer the questions in their “professional capacity about codebases associated with [their] professional work, whether closed-source or open-source.”

It’s worth noting that this survey is not and was not meant to be representative in the statistical sampling sense. All surveys of software developers and software professionals struggle with this same methodological challenge of defining an ill-defined professional category. That said, the next section presents data on the respondents who did take the survey, providing concrete evidence the survey did capture “professionals involved in the creation and maintenance of software.”

Who Took the Survey?

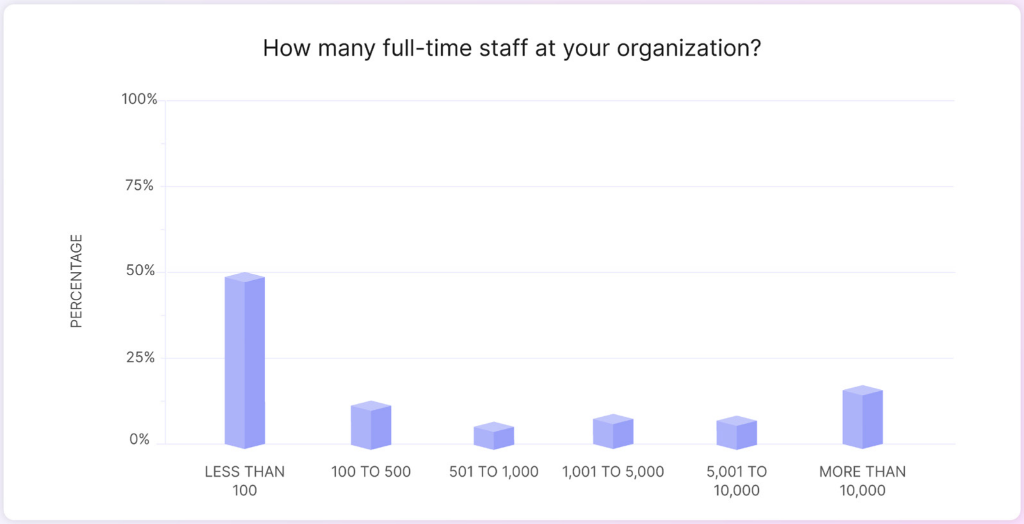
The survey was ultimately completed by 167 respondents. The survey asked each respondent eight questions related to their background. In aggregate, analysis of answers to these questions suggests that the sample was filled with software professionals, especially software developers, from around the world, at organizations of varying size, with a wide variety of security postures. Crucially, the results also indicated these respondents were not particularly familiar with SLSA before the survey and that a majority of respondents worked at organizations that were not yet implementing SLSA, suggesting that this survey did not end up only in the hands of SLSA enthusiasts and supporters.

The survey asked about the professional role of the respondents. Eighty-two of 167 respondents reported being a software engineer/developer and 24 indicated that they were a manager, executive, or director. The remaining responses were distributed over a large number of idiosyncratic job titles.

Respondents also reported the number of full-time staff at their organization. See figure one for the distribution. Nearly fifty percent of respondents worked at organizations with less than 100 full-time staff. The remaining respondents were relatively evenly distributed at larger organizations.

FIGURE 1.

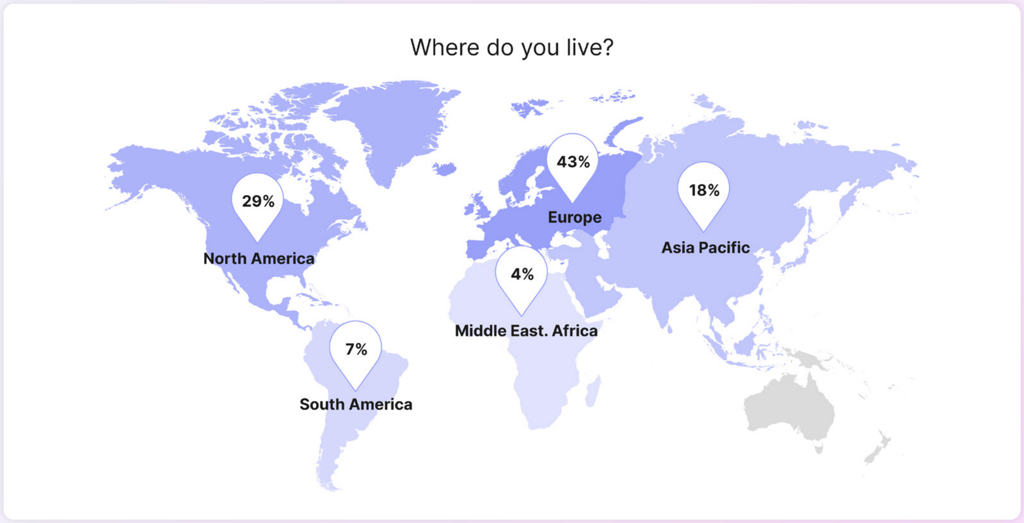
Full-time staff at respondent's organization



To understand the geography of respondents, the survey also asked about which region the respondent lived in. As figure two indicates, over 40 percent of respondents resided in Europe. North American and Asian Pacific-based respondents were the next two most common groups.

FIGURE 2.

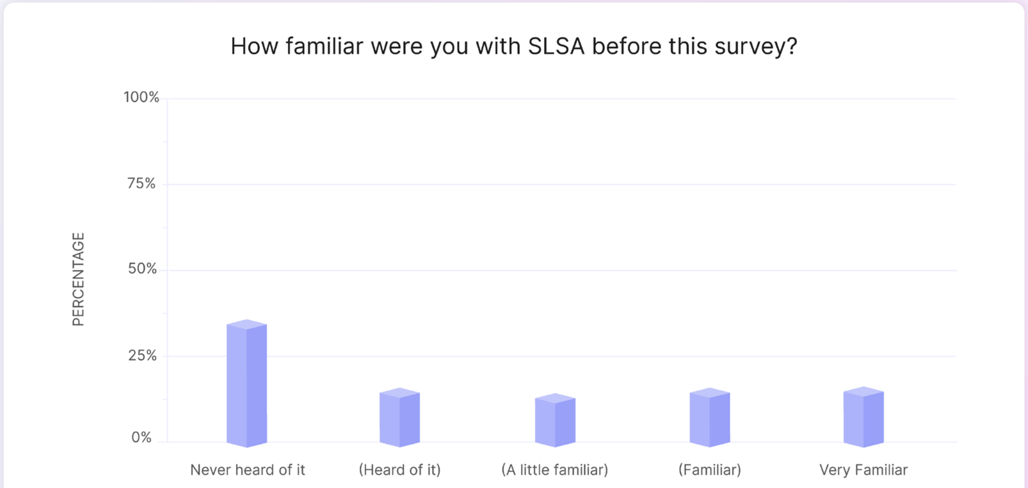
Respondent Geography



Respondents reported a wide range of familiarity with SLSA. Figure three displays the levels of familiarity. Over 30 percent of respondents had "never heard of it" prior to the survey. Figure three does suggest a wide variety of familiarity with SLSA. Additionally, only approximately thirty percent of respondents indicated their organizations currently use SLSA.

FIGURE 3.

Prior Familiarity with SLSA

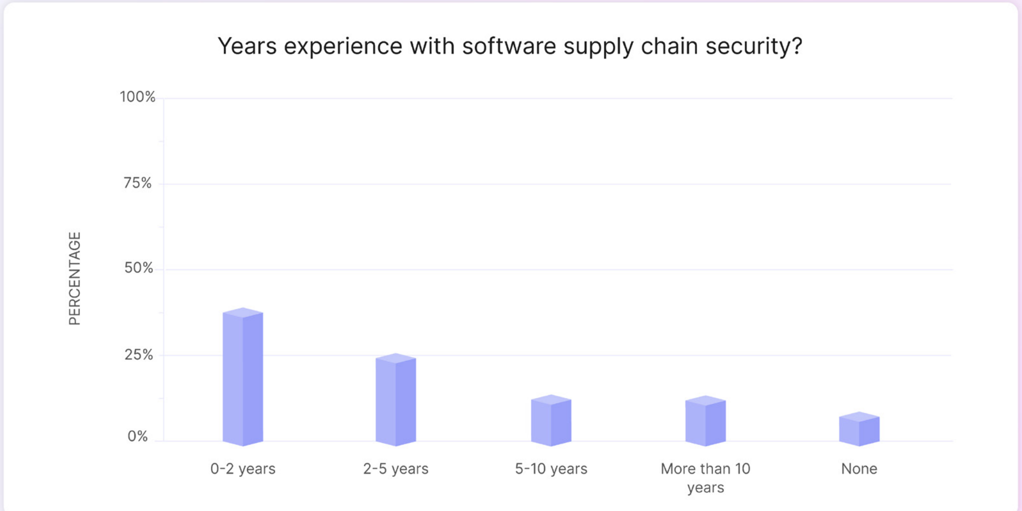


Note: Respondents were provided a 1-5 scale and explicitly instructed that 1 indicated "never heard of it" and that 5 indicated "very familiar." Levels 2-4 were not given an explicit description. In the graph above and in all graphs below, any description surrounded by parentheses, e.g. "(Heard of it)", indicates an implicit label assigned by the research team after the fact.

The survey also requested information on the respondent's years of experience with software supply chain security. Figure four indicates that the majority of respondents had fewer than five years of experience, with the plurality possessing fewer than two years of experience.

FIGURE 4.

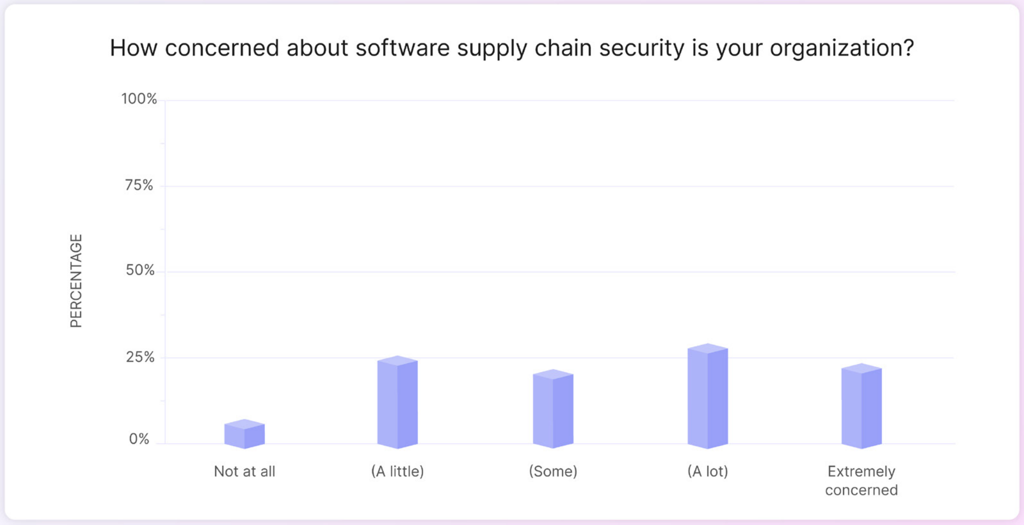
Experience with software supply chain security

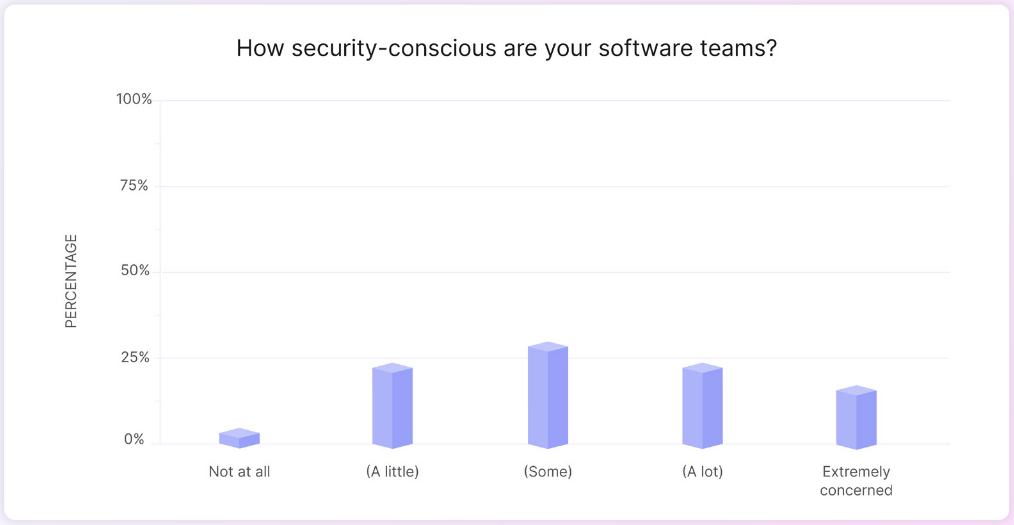


The respondents also reported on the extent to which their organizations are concerned about software supply chain security and how security-conscious the software teams are at their organizations. See figures five and six. Respondents reported a wide range of levels for both questions, indicating that the survey did not only capture particularly security-conscious organizations.

FIGURES 5 AND 6.

Concern about software supply chain security and security at respondent's organization





In sum, the survey captured "professionals involved in the creation and maintenance of software" with a wide variety of exposure to SLSA and software supply chain security.

Findings

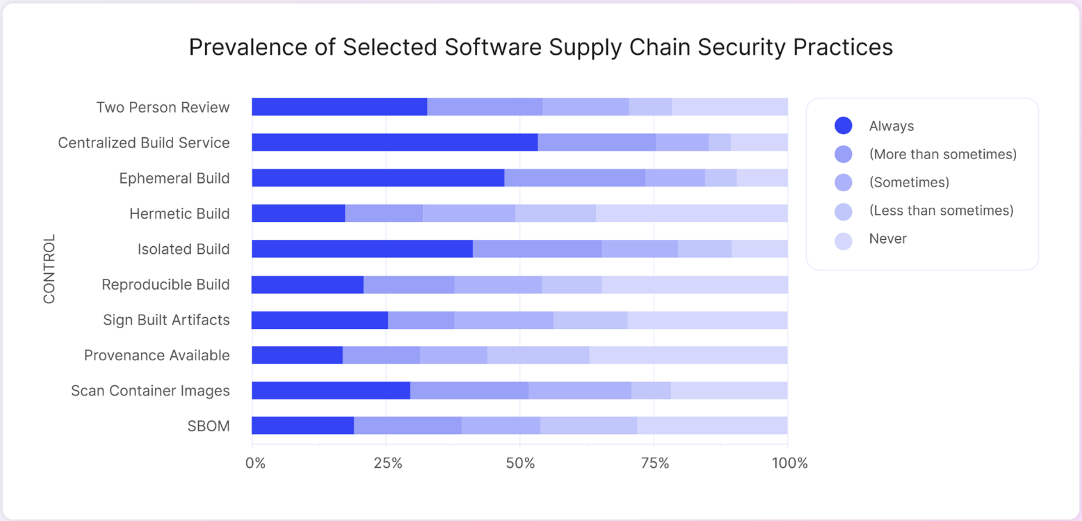
The findings section is divided into three analyses, corresponding to questions related to (1) the prevalence of selected software supply chain security practices, (2) their perceived utility, and (3) their perceived level of difficulty.

1 Some software supply chain security practices are widely practiced, though providing provenance, arguably a key SLSA-related practice, notably lags in adoption.

Some of the SLSA-related practices, as indicated in figure seven, are already widely adopted. The majority of respondents report relatively widespread use of a centralized build service, ephemeral builds, and isolated builds. The second-tier of most commonly used practices includes review by two trusted persons, vulnerability scanning, use of SBOMs, and digital signatures. Finally, reproducible builds, hermetic builds, and making provenance available are, relatively speaking, laggards.

FIGURE 7.

Prevalence of Selected Software Supply Chain Security Practices



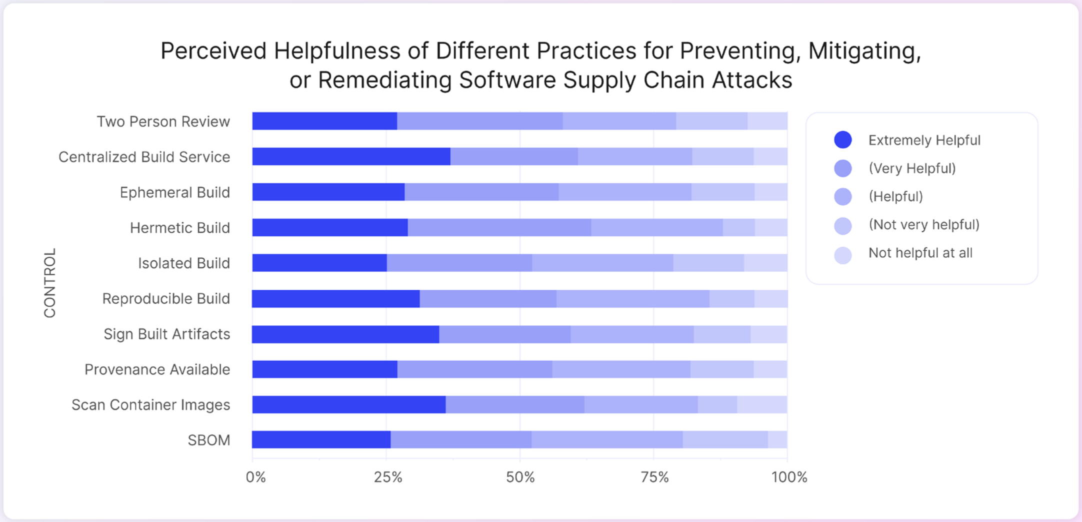
Reassuringly, the pattern of answers is similar to the [2022 State of DevOps report] section on SLSA. For audiences interested in SLSA-related practices, the relative lack of adoption of the provenance requirement is a potentially surprising finding. To put it simply, provenance appears to have noticeably less adoption than using a centralized build system, SBOMs, or vulnerability scanning. Without provenance—verifiable information about software artifacts describing when, where, and how something was produced—achieving the integrity goals of SLSA is, to say the least, difficult.

2 There is hardly any variation in the perceived usefulness of different software supply chain security practices.

According to figure 8, respondents evaluated the helpfulness of different software supply chain security practices as essentially the same. Why? The survey data can't answer this question directly, but one plausible explanation is that software supply chain security is a sufficiently new domain where strong opinions haven't formed yet.

FIGURE 8.

Perceived Helpfulness of Different Practices for Preventing, Mitigating, or Remediating Software Supply Chain Attacks



The relatively low ranking of making provenance available could be related to attitudes similar to the one expressed by one respondent in the free-text response:

> This seems like a way to produce a ton of paperwork and make it easy to go back after the fact and say "oh, here's the attack"... while doing little to prevent compromises from happening in the first place.

Respondents also expressed concerns about high false positive rates when scanning containers for known vulnerabilities. One respondent opined:

> False positive rates are extremely high with the current tooling to the point that the cost per averted vulnerability is quite elevated.

Another offered their frustrating experience with large container base images:

> Our docker images are not that slim, so there is [a] lot of noise from packages in base images, so most findings are not quickly acted on. This works better if docker images can be slimmed down.

On the topic of SBOMs, some respondents expressed concern about the utility of SBOMs. For instance, one survey respondent wrote, "Generating SBOMs is not really difficult anymore, but what do I do with it afterward?" Another offered a perspective that many SBOM advocates will find pessimistic:

> This is the kind of paperwork that is tedious and disliked by everyone: devs (because they have to write up and possibly defend their many random dependencies), management (because this introduces delays and unhappy devs), even legal (because it risks turning accidental infringement into wilful). Still, being mindful of dependencies seems like the only good way to reduce the risk of supply-chain attacks.

Another respondent commented on their belief that SBOMs aren't necessary to enable vulnerability scanning.

> We can get vulnerabilities very easy [sic] without creating a SBOM. The SBOM is there for us if a vendor asks for a SBOM of the solution. So it's good for that purpose, but that request is rare.

Finally, an analysis of the relationship between the perceived utility and the relative adoption found that there was a statistically significant relationship ($p < .01$ using a simple linear regression) between these two variables for all practices covered by the survey.

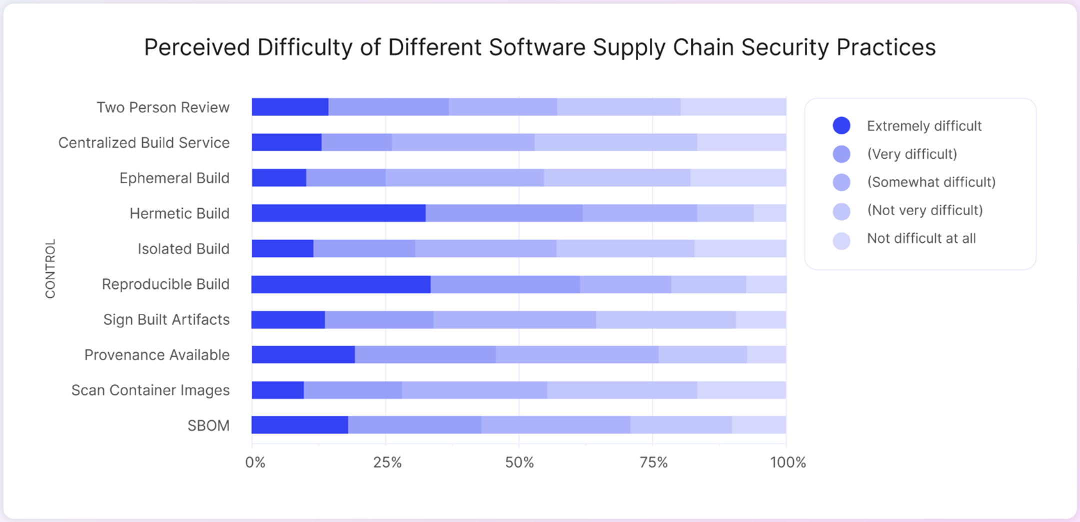
3

Some SLSA practices--especially hermetic and reproducible builds--are considered substantially more difficult than others.

Unlike the perceived helpfulness of software supply chain security practices, the perceived difficulty varies widely. See figure nine. Two SLSA practices, according to respondents, stand out as particularly difficult: hermetic and reproducible builds. One respondent lamented in a free-text response that hermetics builds, due to the widespread use of “internet-based dependencies,” are “extremely difficult.” A number of respondents also pointed out that they or their organization do strive for reproducibility, but not necessarily bit-for-bit identical output. Interestingly, making the provenance document available clocks in as the third most difficult practice.

FIGURE 9.

Perceived Difficulty of Different Software Supply Chain Security Practices



While an observer might expect that the wide variation in perceived difficulty explains relative patterns of adoptions, a detailed analysis suggests otherwise. No individual practice exhibited a statistically significant ($p < .05$ using a simple linear regression) between perceived difficulty and relative adoption.

The recent decision within the SLSA community to defer consideration of hermetic and reproducible requirements to after version 1.0 is consistent with the survey’s finding that these requirements are particularly challenging.

Conclusion

So what do the survey results say about software supply chain security?

First, software supply chain security is not merely a distant hope to be enjoyed in a far off future. Many of the practices associated with SLSA already enjoy wide adoption, though there is room for improvement. Second, most practices are viewed as helpful, though this perceived utility hardly varies (in the aggregate) while the perceived difficulty of different practices varies widely (in the aggregate). That said, detailed statistical analysis finds that current patterns of adoption are better explained by perceived usefulness rather than perceived difficulty. Third, supporters of SLSA should take heed of a couple of findings. That making provenance available was the least prevalent of the different practices and was considered among the most difficult might surprise supporters who view making provenance available as straightforward.

Software supply chain security need not simply be a LinkedIn meme or a breaking news story when there is a widespread compromise. This survey, in fact, suggests that it is not, that some of these practices can already be found among some software organizations. The next step is understanding how to make these and related practices the default option and, ideally, so commonplace as to no longer merit meme status.

Resources:

- Supply Chain Levels for Software Artifacts, https://slsa.dev/
- Open Source Software Supply Chain Best Practices at the Eclipse Foundation, Eclipse Foundation, https://github.com/eclipse-cbi/best-practices/blob/main/software-supply-chain/osssc-best-practices.md, accessed February 10, 2023
- Rust Foundation, Annual Report 2022, available at https://foundation.rust-lang.org/news/rust-foundation-annual-report-2022/
- Brian Behlendorf, OpenSSF Year in Review, 2022, available at https://openssf.org/blog/2022/12/29/openssf-year-in-review/
- Kim Lewandowski, Building trust in our software supply chains with SLSA, 2022, available at https://www.chainguard.dev/unchained/building-trust-in-our-software-supply-chains-with-slsa
- Marcus Meissner, Jana Jaeger, SLSA: Securing the Software Supply Chain, 2022, available at https://documentation.suse.com/sbp/server-linux/html/SBP-SLSA4/index.html
- Derek DeBellis, Claire Peters, Accelerate State of DevOps 2022, available at https://cloud.google.com/blog/products/devops-sre/dora-2022-accelerate-state-of-devops-report-now-out