THE LINUX FOUNDATION | Research

Maintainer Perspectives on Open Source Software Security

In partnership with

OpenSSF
OPEN SOURCE SECURITY FOUNDATION

Survey-based Insights from Maintainers Regarding How They Address Best Practices for Secure Software Development

Stephen Hendrick, *The Linux Foundation*
Ashwin Ramaswami, *The Linux Foundation*

Foreword by Stephen Augustus, *Cisco*

December 2023

Maintainer Perspectives on Open Source Software Security

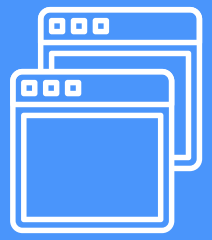
By the end of 2023, 72% of maintainers and core contributors feel that **OSS will be secure.**

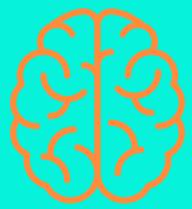
The #1 approach to evaluating the security of OSS packages in use are **SCA and SAST security tools.**

39% of maintainers and core contributors **manually review** source code.

Project documentation is widespread but not ubiquitous: 87% of projects responded that they provide basic documentation.

56% of projects **support reproducible builds.**





Making security tools more intelligent is the #1 approach to improve security across the OSS supply chain.

Reducing developer fatigue through **automation** is the #2 approach to improve security across the OSS supply chain.

69% of OSS contributors want **defined best practices** for secure software development.

The #1 reason for maintaining OSS projects is the **enjoyment of learning.**

49% of OSS contributors want **employer incentives** for OSS contributions.

One-quarter (27%) of maintainers are responsible for **defining OSS security policy.**

30% of maintainers are responsible for **implementing OSS security policy.**

Contents

Foreword

Over the past few years, we have become increasingly more accustomed to conversations around software supply chain security.

If you're about to take a deep dive into these insights presented by LF Research, in partnership with the Open Source Security Foundation (OpenSSF), you're no doubt familiar with the principle that open source software permeates the vast majority of the software that is built today. With that in mind, it stands to reason that secure software development is a practice that must include considerations around how to cultivate secure open source software projects and communities.

I am not an expert in secure software development. I do, however, hold a few perspectives in my daily work: that of a consumer, a contributor, a maintainer, and a sponsor of open source projects.

As consumers, we are concerned with the validity and viability of a project. Does this project do what I need and expect it to do? Is it easy to deploy? Is it written in a language that my team can readily reason about?

As contributors, maybe we are drawn to a project because of our work commitments or maybe our personal interests in the problem space and execution. With this lens, we want to understand the needs of the project, how its community works, and how to become a valued and effective member of that ecosystem.

As sponsors, we trend towards investing in areas that have material impact for our businesses. We should endeavor to support projects not with mandates that only give attention to areas of development that are important to us, but through balanced initiatives that affect the long-term sustainability of the project, its personnel, and the wider ecosystem.

With these lenses in mind, we as maintainers have a multi-directional role. We're dedicated to making our projects a delight to consume. We want to engender a genuine interest in contributing learnings back to the project. We yearn to create an environment in which contributors are supported in becoming maintainers. Finally, we hope that all of what we've built and sustained is an attractive enough value proposition for sponsors to support.

To generalize what I stated before, we are not all experts in secure software development. From this maintainer's perspective, as the report from LF Research highlights, we need to be invested in a few critical areas: educating ourselves on secure software development practices, implementing and leveraging tools that can allow us to more readily produce secure code, and creating feedback loops with other open source practitioners to discuss, improve, and evangelize secure best practices.

In contemplating these maintainer perspectives, I encourage you to examine how this research can catalyze your own role in securing open source software.

After all, open source is a group activity.

STEPHEN AUGUSTUS
HEAD OF OPEN SOURCE, *CISCO*

Introduction

More than 90% of organizations worldwide use open source software (OSS).[1] OSS is present in the entire software stack, from operating systems to infrastructure, middleware, data management, services, frameworks, components, and applications.

OSS maintainers are responsible for managing the development and ongoing upkeep of open source components. Maintainers occupy a crucial role in the OSS ecosystem in steering the direction of OSS projects and ensuring their health and sustainability. Open source relies on its people: its communities of maintainers and other contributors who do tasks ranging from designing features to writing documentation, fixing bugs, and reviewing code. These tasks are critical to supporting and securing the open source security ecosystem. But to be effective and sustainable in the long run, such efforts must ultimately support and empower maintainers, not add additional burdens. Tools, practices, and initiatives around security must be easy to adopt and help empower maintainers in the open source community.

To this end, Linux Foundation Research has surveyed the security of the OSS supply chain. The survey focuses on understanding perspectives on OSS security and the uptake and adoption of security best practices by maintainers, core contributors, end users, and other members of the OSS ecosystem. This survey included questions about secure software development that were answered specifically by OSS maintainers and core contributors only. The survey took place in March 2022, with some survey results published in June 2022. This paper presents previously unpublished information on the adoption of best practices for secure software development from that survey. For more information about this research approach and sample demographics, see the methodology section of this paper.

1. Adrienn Lawson and Stephen Hendrick, World of Open Source: Global Spotlight 2023 (San Francisco, The Linux Foundation, 2023), 9.

Secure development best practices is an area where the Linux Foundation, and specifically the Open Source Security Foundation (OpenSSF), have established best practices for secure software development—**Best Practices Badge (bestpractices.dev)** & Scorecard (**securityscorecards.dev**)—and provide free training and certification in secure software development (**Developing Secure Software (LFD121)—Linux Foundation— Training**).

Security challenges

Addressing the security of OSS components requires a different approach from traditional approaches to securing proprietary, vendor-supported software. The more loosely structured and community-focused nature of typical OSS development presents a different environment for addressing software security where there are a few large visible projects (such as the Linux kernel and Kubernetes) and many small projects define the distribution of OSS projects. Smaller projects typically have fewer contributors and resources and are therefore more likely to adopt a minimalist approach to development and security.

The tremendous benefits and prevalence of OSS in organizational software, combined with vulnerabilities in the OSS software supply chain, put us at a crossroads. Organizations and companies that use OSS need to become more aware of what dependencies they are using, proactively and regularly monitoring all components for usability, trustworthiness, and vulnerabilities. Ultimately, OSS is a two-way street. Consumers of OSS must contribute back to the OSS communities to ensure the health and viability of the dependencies they rely on. Merely using OSS without contributing back is not enough if its users want to ensure that the software will meet their needs over time and requires them to (a) incorporate the nature of OSS dependencies into standard cybersecurity and development practices and (b) contribute back to the OSS communities that organizations rely on.

Comparing perspectives of open source maintainers to other open source software contributors

Maintainers and core contributors represent 36% of open source contributors

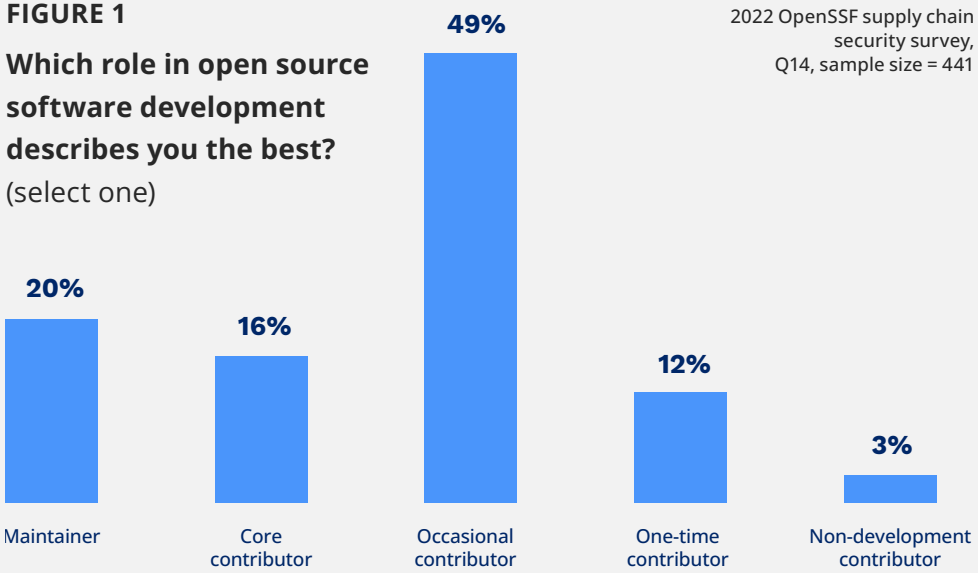
Our open source supply chain survey included 441 respondents that self-identified as OSS contributors. The distribution of these OSS contributors shown in Figure 1 is as follows: maintainers (20%), core contributors (16%), occasional contributors (49%), one-time contributors (12%), and non-development contributors (3%). Definitions for each of these roles are as follows:

- **Maintainer:** A software maintainer or package maintainer is the final decision maker over all or portions of source code that goes into a build or release. Maintainers would likely also identify as a subset of core contributors.
- **Core contributor:** A core contributor may have been part of the project since inception or joined later, regularly participates in major discussions about project direction, and has significant ongoing roles in the work, possibly including accepting patches to the code base. A project community may refer to core contributors as "Committers."
- **Occasional contributor:** An occasional contributor would not normally participate in ongoing or weekly project discussions but occasionally provide contributions over longer time periods.
- **One-time contributor:** A one-time contributor is someone who provides a specific set of suggestions or contributions and then exits involvement once their work is done. These are sometimes called "drive-by commits."
- **Non-developmental contributors:** These are other IT staff with a strong focus on software security.

FIGURE 1

Which role in open source software development describes you the best?
(select one)

2022 OpenSSF supply chain security survey, Q14, sample size = 441

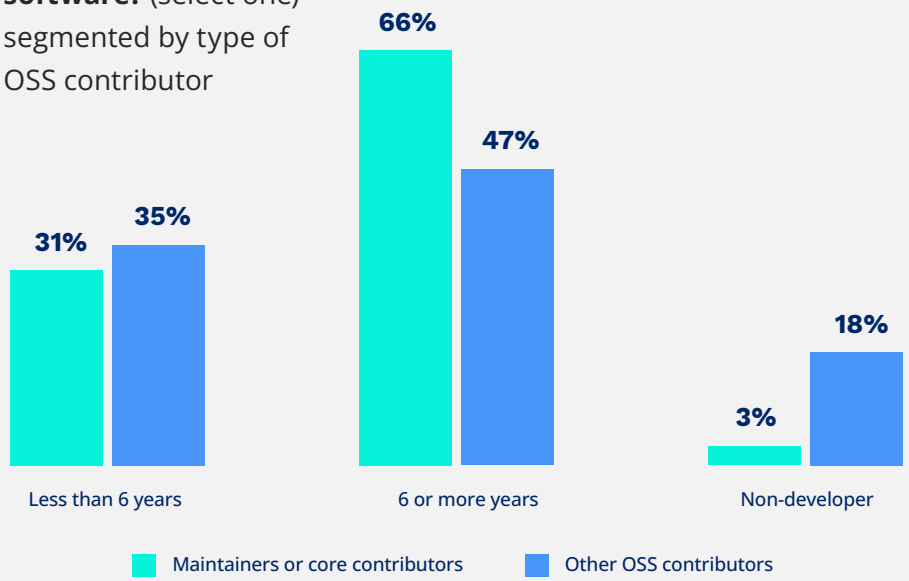


We will use Figure 1 to segment OSS contributors into two categories: maintainers and core contributors (36%) and other OSS contributors [occasional contributors, one-time contributors, and non-development contributors (64%)]. These two OSS segments represent those contributors with a high level of OSS contribution or involvement (maintainers and core contributors) and those with a passing or low-level contribution or involvement (other OSS contributors). Throughout this report, we will compare the actions, beliefs, and perceptions of these two segments to see where they are the same and where they are different.

FIGURE 2

How long have you developing open source software? (select one) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q13 x Q14a, sample size = 441



Maintainers and core contributors have an experience advantage

While the demographics across the maintainers and core contributors segment compared to the other OSS contributors segment are similar across most metrics (age, employment, geography, company size, industry, role, and area of responsibility), the one metric where there is a difference is years of experience. Figure 2 shows that 66% of maintainers and core contributors have six or more years of experience compared to 47% for other OSS contributors.

The share of maintainers and core contributors who are non-developers is also very low at 3% compared to other OSS contributors at 18%. This is understandable because the role of maintainers and core contributors is often to contribute source code, and even maintainers who do not directly write code must make decisions about what contributions should receive approval (including code) and to document or inform the community about the new changes. It would be hard for a maintainer to understand what changes to code should receive acceptance without understanding code.

Open source contributors are concerned about open source software security, but not as much as you might expect

When asked about how secure their process is for developing or using OSS today, Figure 3 shows that most maintainers and core contributors (62%) and other OSS contributors (63%) view the process for developing or using OSS code as secure. This contrasts with the 19% of OSS contributors who view this process as insecure and the 17 to 19% of OSS contributors who remain neutral. The data in this report delivers a more nuanced picture of software development security and points to the methods used by OSS contributors to address software security and the differences between the actions and expectations of these two respondent segments: maintainers and core contributors and other OSS contributors.

Open source contributors have questionable expectations about how open source security will improve

Given the baseline of 62% to 63% of OSS contributors who stated in Figure 3 that OSS software in development or use was secure in 2022, it is remarkable to see the optimism that OSS contributors have in how the state of OSS would change by the end of 2022. Figure 4 shows that 68% of maintainers and core contributors believe that OSS software in development or use would qualify as secure by the end of 2022 (up from 62% in Figure 3), and just 8% believe it would be insecure (down from 19% in Figure 3). This leaves 24% who are neutral (neither secure nor insecure). Maintainers and core contributors only expect modest gains in software security by the end of 2022, but the significant decline in maintainers and core contributors who feel that OSS

FIGURE 3

How secure is your process for developing or using open source software today? (select one) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q17 x Q14a, sample size = 348

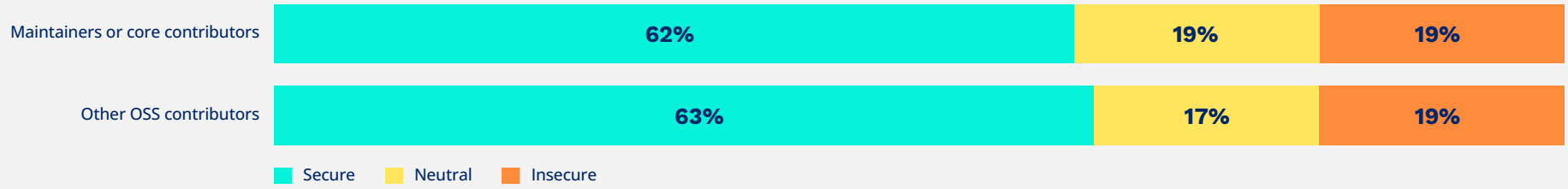
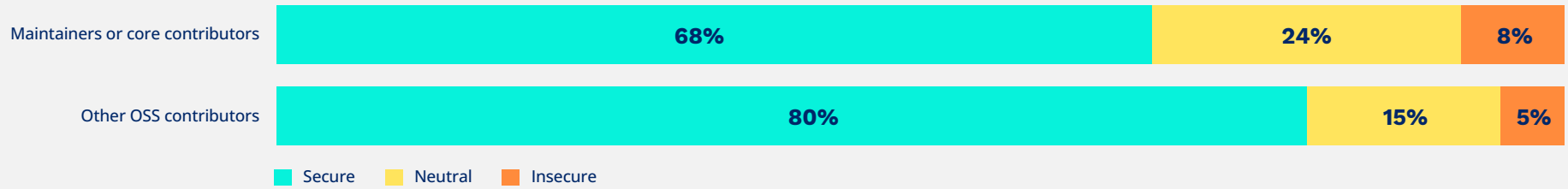


FIGURE 4

How do you see the security of open source software you develop or use changing in 2022? (select one) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q18 x Q14a, sample size = 348



software remains insecure drops dramatically from 19% early in 2022 to 8% by the end of 2022.

The drift for other OSS contributors is even more extreme. Figure 4 shows that 80% of other OSS contributors believe that OSS software in development or use would qualify as secure by the end of 2022 (up from 63% in Figure 3), and just 5% believe it would be insecure (down from 19% in Figure 3).

We then asked yet another follow-up question regarding how the security of OSS would evolve in 2023. This is where the responses diverged even more. Figure 5 shows that 72% of maintainers and core contributors believed that OSS would be secure, which is up four points from the late 2022 (Figure 4) value and up 10 points from early 2022 (Figure 3). This increase from 62% in early 2022 to 72% by the end of 2023 suggests that maintainers expect to see a moderate improvement over this two-year period. The following developments can explain this type of improvement over two years:

- Improved security testing tools, including static application security testing (SAST) and dynamic application security testing (DAST) accompanied by machine learning (ML) to detect security issues
- Higher levels of community involvement
- Better dependency management
- An increased focus on DevSecOps
- Government involvement and regulation
- More emphasis on secure software development in developer training

However, other OSS contributors appear convinced that these types of software security improvements will have an even more significant impact. Figure 5 shows that 87% of other OSS contributors feel that OSS use and development will be secure by the end of 2023, up seven percentage points from the end of 2022 (Figure 4) and up 24 points from early 2022 (Figure 3). This data shows a widening gap between the perspectives of maintainers and core contributors compared to other OSS

FIGURE 5

How do you see the security of open source software you develop or use changing in 2023?
(select one) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q19 x Q14a, sample size = 348

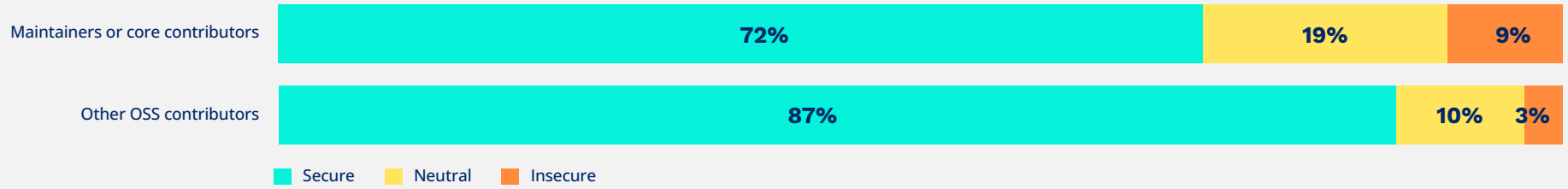
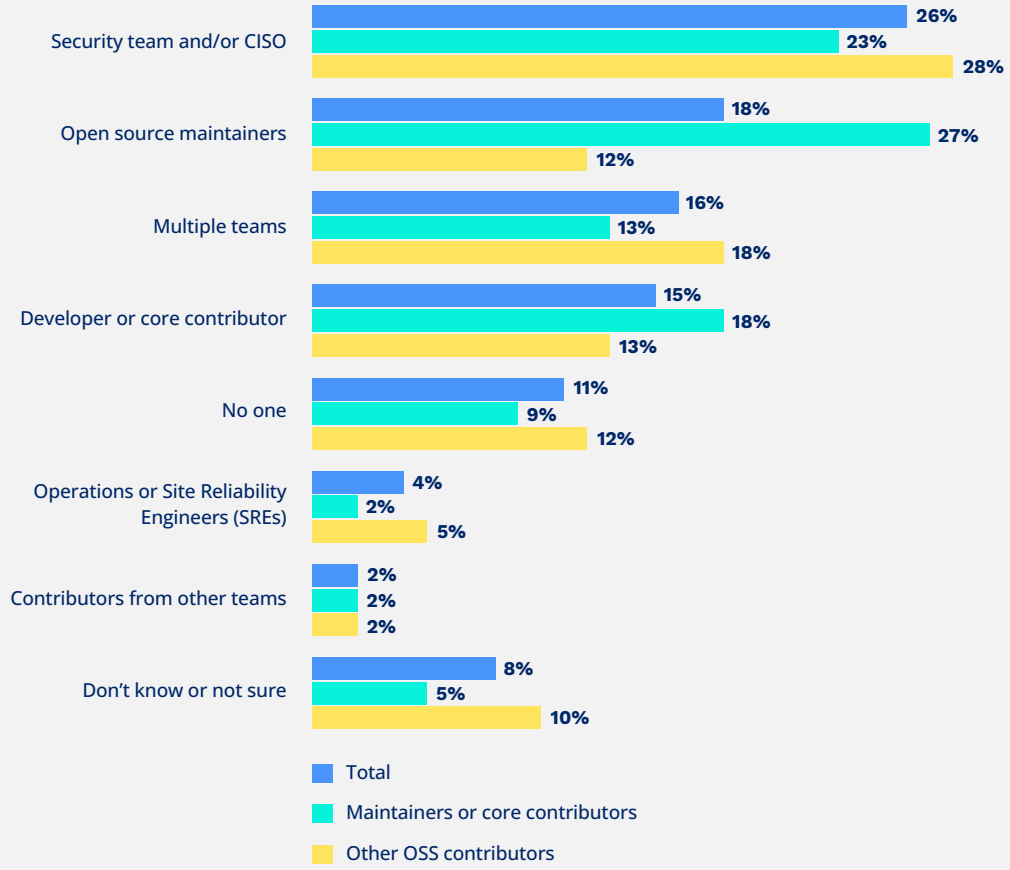


FIGURE 6

Who is responsible for defining your OSS security policy?

(select one) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q21 x Q14a, sample size = 348



contributors. Our discussions with maintainers suggest that these dramatically differing perspectives are due to experience. Because other OSS contributors are only participating peripherally in maintaining OSS, they are likely to oversimplify the required steps to improve OSS security. When your involvement in an activity is not deep, it often appears to be much easier than it is, which is an example of the Dunning-Kruger effect.

Open source contributors are involved in setting open source security policy

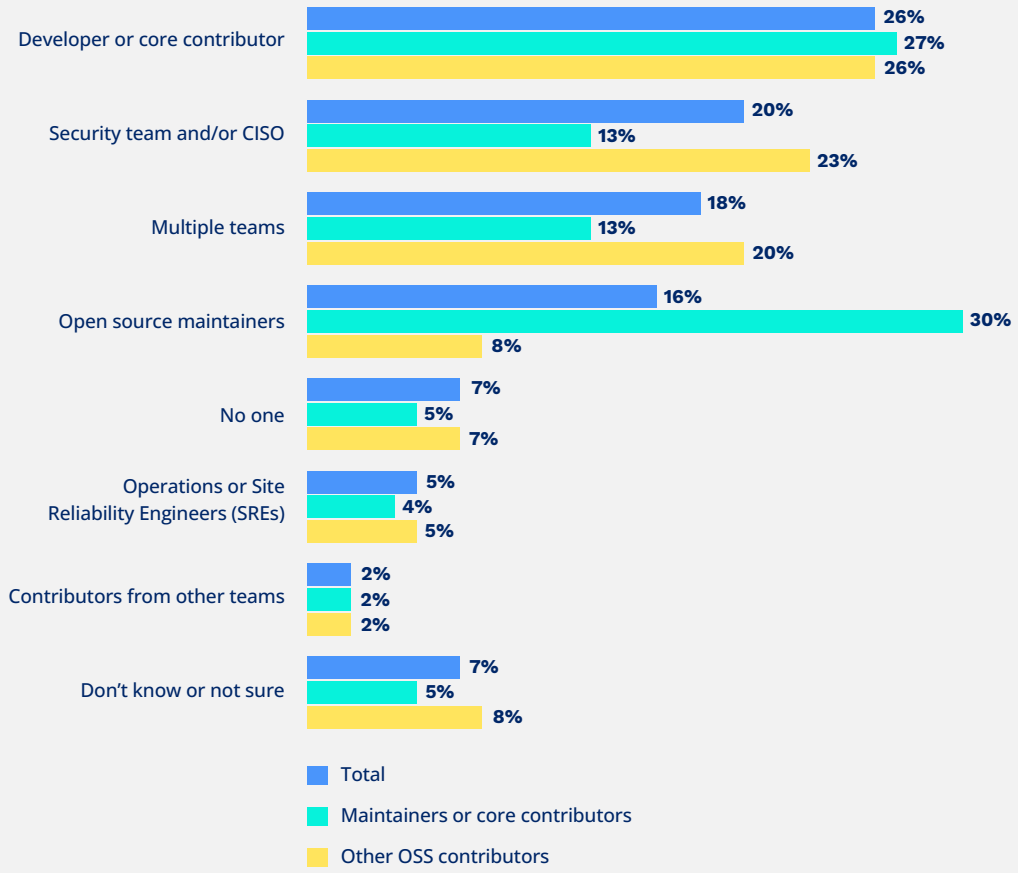
Although Figure 6 reflects the perspective of OSS contributors, it showcases the high level of involvement that OSS contributors have in setting OSS security policy. Overall, Figure 6 shows that the primary approach to defining OSS security policy within the organization is the responsibility of the CISO and / or security team. However, 27% of OSS maintainers (and 12% of other OSS contributors) identify maintainers as having responsibility for setting OSS security policy. This is likely to be the position that many organizations take where there isn't a security team, CISO, or open source program office (OSPO) claiming the role.

Figure 6 also shows that involving multiple IT teams to define OSS security policy is sometimes a solution. The reason for this is that cybersecurity needs are far-reaching, involving developers, network security, risk management, compliance, and more. Leveraging multiple teams can bring diverse experiences to bear and enable the inclusion of more aspects of cybersecurity in the policy defined.

FIGURE 7

Who is primarily responsible for Implementing security across development and usage activities?
(select one) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q22 x Q14a, sample size = 348



Open source contributors and developers are all tasked with implementing security policy

Figure 7 shows that when we asked who was primarily responsible for implementing security policy, developers and core contributors were the overall leading response (all OSS contributors) at 26%. Security teams and / or the CISO (20%), multiple teams (18%), and open source maintainers (16%) reinforce the role that developers have while also identifying the importance of security teams in implementing security policy.

What is somewhat surprising is the difference in beliefs that emerge between maintainers and core contributors and other OSS contributors in three of the responses in Figure 7: security teams and / or CISO, multiple teams, and open source maintainers. Each segment's experience and frame of reference appear to impact the disparity between segments. However, we assess that cybersecurity policy implementation takes a village because of its wide scope, and while it is natural to expect the involvement of maintainers and core contributors, this will also include other developers and security teams.

Maintainers demonstrate the importance of manual inspection to address cybersecurity

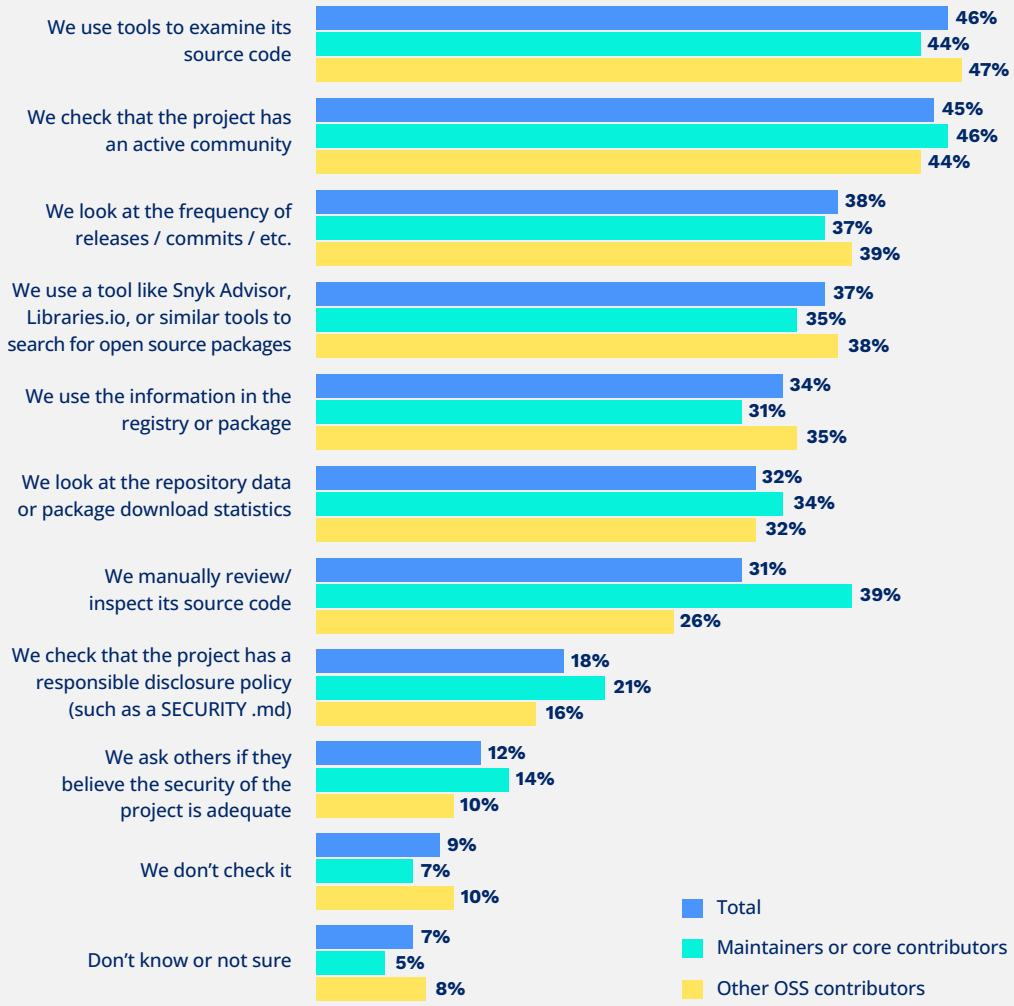
The most common approach to checking the security of OSS packages that are and will be in use is to utilize tools to examine the source code and investigate its activity. Figure 8 shows that 46% of OSS contributors use tools to introspect source code; 45% also check to make sure that the project has an active developer community; and 38% look at the frequency of releases and commits. It is also important to note the consistency of both OSS contributor segments in their support for these leading approaches to source code introspection.

FIGURE 8

How do you check the security of the open source packages that you use?

(select all that apply) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q28 x Q14a, sample size 348, valid cases = 348, total mentions = 1,070



Using tools to examine source code is the leading approach for a good reason. Developers utilize tools because they have limited time. Tools provide a faster way to introspect code. Machine learning is likely to eventually improve the capability of security tools, but the consensus seems to be that there is still no substitute for manual inspection.

There is one significant difference in Figure 8, which is that 39% of maintainers and core contributors manually review and inspect source code. The primary role of maintainers and core contributors is to improve component functionality; however, you can't modify code unless you have a good idea about how it works. Therefore, maintainers do need to manually review code to understand the delivery of its functionality. Addressing the security of OSS components may be a cumbersome side effect of maintaining code, but manual code review facilitates it. So, maintainers do prioritize manual code reviews and inspections to improve the functionality and quality of the code they change or add, but they also use this opportunity to identify and address security concerns.

Security tools use is correlated with the value provided

Figure 9 shows that 50% of all OSS contributors use software composition analysis (SCA) tools. SCA tools are very effective at identifying license compliance issues and common vulnerabilities and exposures (CVEs) across OSS components in use by your organization. SCA tools are not difficult to integrate into a CI / CD pipeline, which has helped with their adoption.

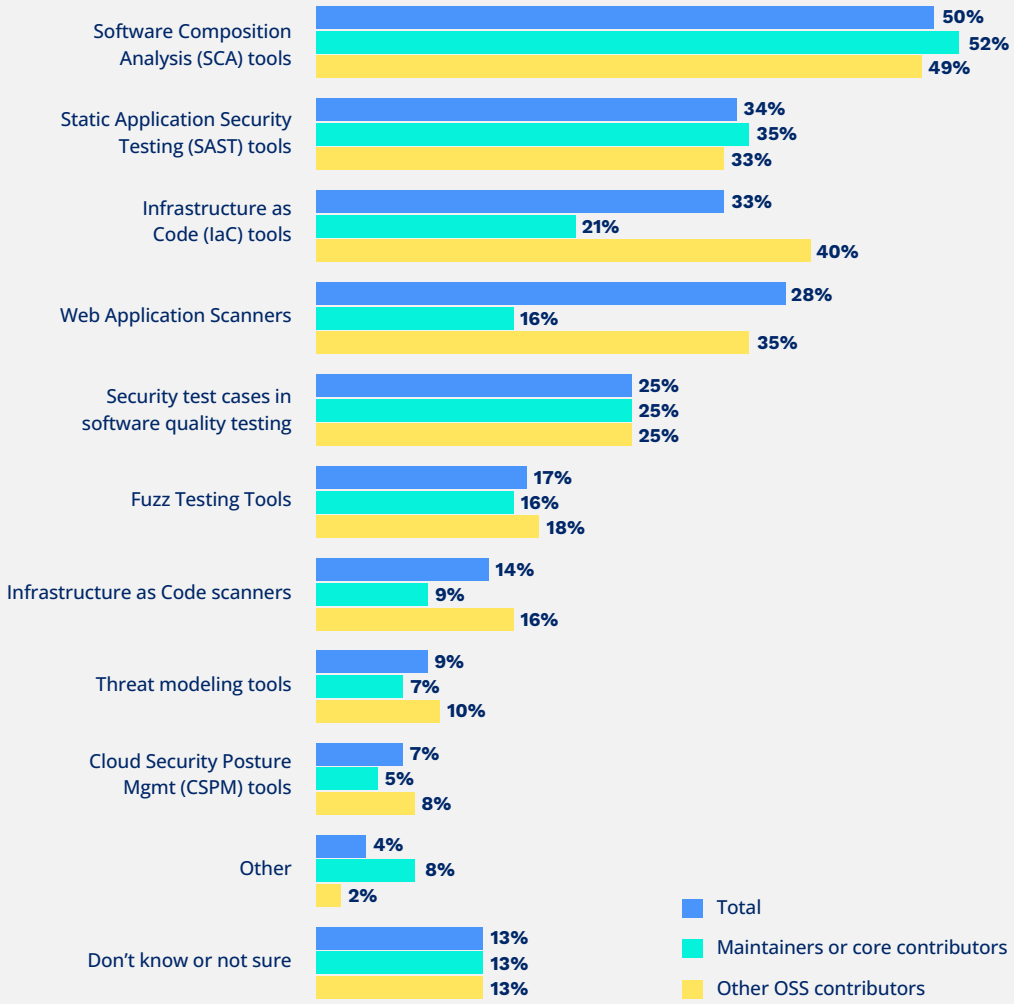
Figure 9 shows the use of SAST tools at 34% and infrastructure as code (IaC) tools at 33%, defining a 2nd tranche of tool use by OSS contributors. The common use of SAST tools, much like SCA tools, in software development is to test code during development, before code is committed, during code reviews, after code is comm-

FIGURE 9

What security tools do you regularly use when developing open source software?

(select all that apply) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q29 x Q14a, sample size 348, valid cases = 348, total mentions = 813



itted, during continuous integration, and before deployment. Finding security vulnerabilities—such as finding bugs—should happen as early in software development as possible. Vulnerabilities and bugs become exponentially more expensive to fix when found later during development. As SAST tools become more intelligent, it would be reassuring to see their adoption increase significantly because of their role as an automated primary defense against security threats.

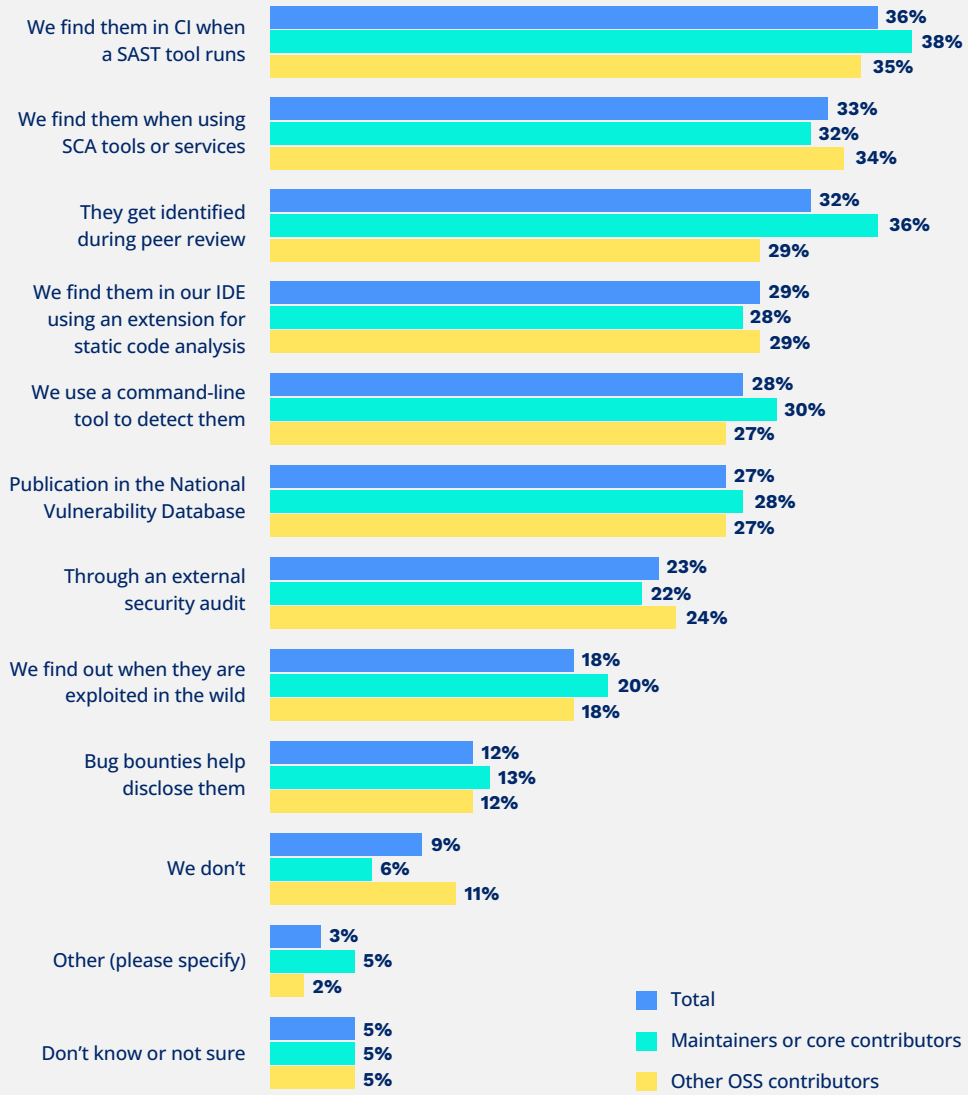
Finally, IaC tools, which 33% of OSS contributors use, are very effective at automating key software development processes. Reducing manual touch points across CI / CD activities is an important way to reduce the exposure of mission-critical activities such as provisioning, management, and deployment through automation. We do note that just 21% of maintainers and core contributors use IaC tools compared to 40% of other OSS contributors. While we do know that maintainers and core contributors are responsible for releasing code for eventual use in a production environment, the role of other OSS contributors is somewhat more ambiguous. What we do know from the data is that other OSS contributors participate more in using IaC tools (40% vs. 21%), web application scanners (35% vs. 16%), and IaC scanners (16% vs. 9%). What we can say is that other OSS contributors are on average more committed tool users. This may be because they have less experience and / or because their roles involve them in more activities across the software development life cycle (SDLC).

FIGURE 10

How do you find out about security vulnerabilities in your code?

(select all that apply) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q30 x Q14a, sample size = 348, valid cases = 348, total mentions = 892



How open source contributors find security vulnerabilities

No one mechanism guarantees to find all vulnerabilities. Therefore, discovering vulnerabilities requires that OSS contributors use a variety of approaches, as Figure 10 demonstrates. They commonly use security tools such as SAST (36%) and SCA (33%), and, in some cases, integrated development environments (IDEs) with SAST extensions (29%), as well as command line tools (28%). However, identifying vulnerabilities during code reviews by OSS contributors (32%) is common and is a primary way for maintainers and core contributors (36%) to find vulnerabilities. We should not underestimate the importance of this approach because it can identify issues including and beyond the scope of just vulnerabilities, despite being more labor-intensive.

SCA tools are a primary way for OSS contributors and users to find known vulnerabilities in reused components. SAST tools can identify unknown vulnerabilities. This is why the use of both SCA and SAST tools is so important.

Figure 10 shows that 36% of OSS contributors use SAST tools, and 33% use SCA tools. Overall, 32% report that they find vulnerabilities during peer review, and once again, maintainers and core contributors (36%) rely more on this approach than other OSS contributors do (29%).

Maintainer perspectives on secure software development

The OSS security survey included a series of questions for only OSS maintainers and core contributors, which allowed them to explain how they performed their maintainer and development responsibilities. As we saw earlier, Figure 1 shows that 36% of OSS contributors are maintainers or core contributors (159 respondents). Of these maintainers and core contributors, 72 were willing to answer questions about the primary open source project they participated in and how they performed development and maintainer responsibilities.

The survey asked a series of seven questions spanning 52 responses about maintainer or core contributor adoption of secure software development best practices. The seven questions we asked comprised OSS development activities selected from project phases, including project management, source code management, the build process, software quality assurance (SQA), software security, security testing, and secure coding.

For each of the 52 best practices (secure software development activities), respondents were able to choose from five stages of adoption:

1. In use now
2. Planned for 2022 or 2023
3. No plans to use
4. Not applicable
5. Don't know or not sure

Please note that if you regard the logistics curve as a proxy for market adoption of a particular product, technology, or best practice, consider 85 to 90% as the "maximum target adoption." The reason for this is that best practices are not always applicable to all maintainers and core contributors. There is always a percentage that doesn't know or is not sure if the development processes in place require or support the best practices. Consequently, any combination of current and planned use that approaches or exceeds this maximum target adoption range is an excellent finding.

Most maintainers and core contributors support basic requirements for OSS use and contribution

We started with the adoption of basic best practices. Figure 11 shows that activities that are now widely in use include ensuring a project has basic documentation (87%), posting the terms of its license (84%), actively maintaining the project (83%), describing what the project does on the project website (80%), and making sure the project enables people to discuss changes and issues (79%).

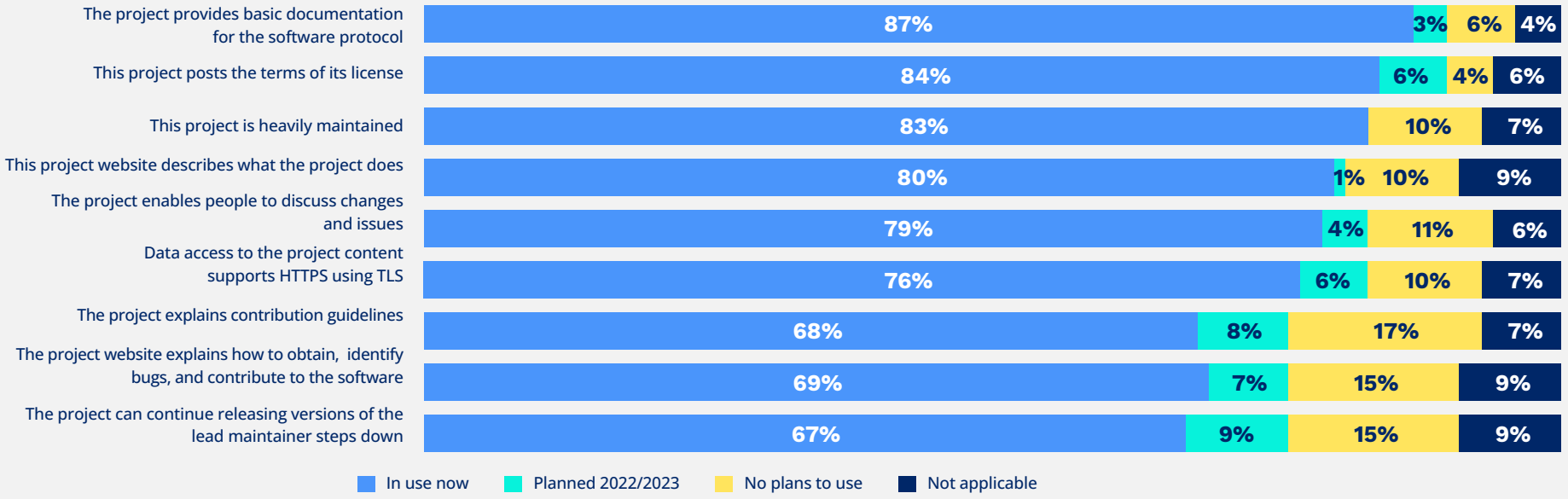
However, Figure 11 also shows that while 87% of maintainers and core contributors said their project provides basic documentation for the software produced, only 68% of these respondents said their project explains contribution guidelines. Fewer maintainers expressed confidence that the project could continue releasing versions if the lead maintainer stepped down. In our qualitative interviews, maintainers noted the fact that sustainability is also a component of software supply chain security. What happens if a maintainer steps down?

A positive characteristic of Figure 11 is that the high percentage of adoption (in use now) diminishes the percentage of maintainers and core contributors who are in a different stage of adoption. Generally, as the number of maintainers or core contributors who use a best practice increases, there will be a decrease in planned

FIGURE 11

Does this project support the following basic best practices?

2022 OpenSSF supply chain security survey, Q38, sample size = 72, DKNS excluded from the analysis



adoption or other adoption alternatives by maintainers and core contributors. We do see this pattern of involvement across the first six best practices. Equally disappointing is the elevated level of maintainers and core contributors in the last three activities who have no plans to address how the project explains contribution guidelines (17%), how to identify bugs and contribute to the software (15%), or what happens when the lead maintainer steps down (15%).

How maintainers approach source code management and change control

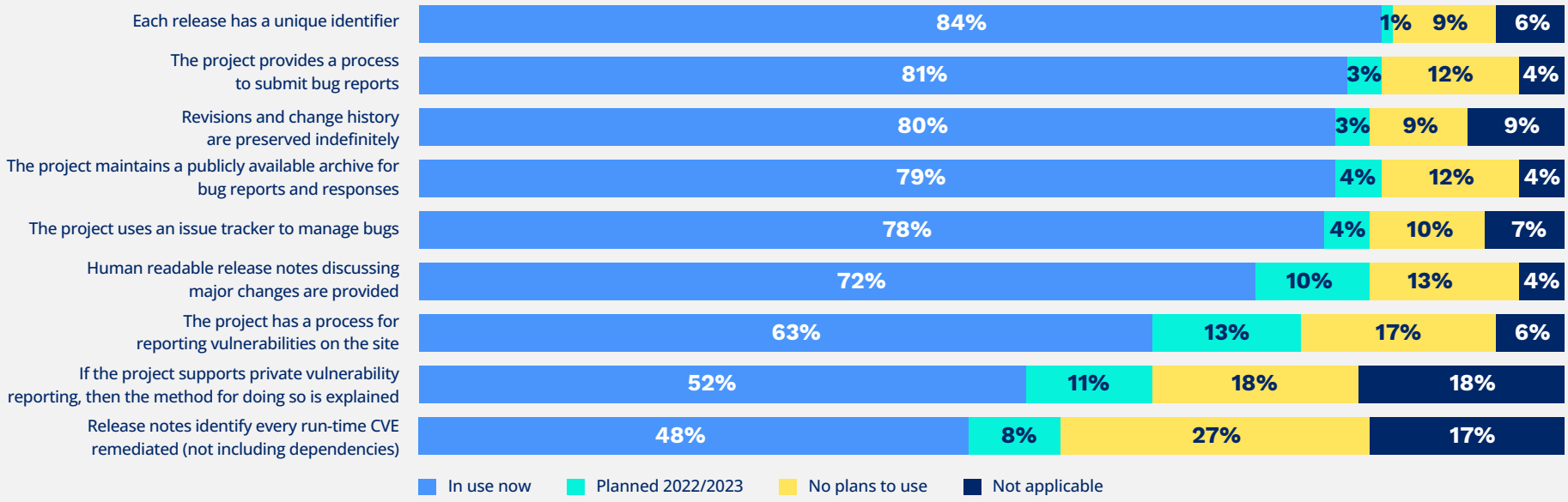
Source code management and change control are fundamental for the effective management, collaboration, stability, and growth of open source projects. They provide the infrastructure and processes necessary for open source communities to collaborate and establish development processes.

Figure 12 shows that 72 to 84% of maintainers and core contributors follow key source code management and change control best practices; however, vulnerability identification and remediation reporting lag considerably, with maintainer and core contributor adoption ranging from 48 to 63%. In our interviews

FIGURE 12

Does this project support the following source code management and change control best practices?

2022 OpenSSF supply chain security survey, Q39, sample size = 72, DKNS excluded from the analysis



with maintainers, some mentioned that they may also get the word out about publicly-known vulnerabilities through more informal channels such as Twitter, email, or Slack, but there should always be a system of record around the management of vulnerabilities by the project, which needs to involve CVE registries and additional informal channels for identifying other vulnerabilities.

Support for build process best practices shows a combination of strengths and weaknesses

This section concerns best practices around the build process. Figure 13 shows that the majority of maintainers and core contributors have adopted best practices to address common build activities such as having build definitions stored in a version control system (VCS) (78%), having all build steps as part of a build script (75%), running the build service in an isolated environment (69%), having the build service run as an ephemeral environment (63%), having dependencies listed in a computer-readable way (63%), using secure design principles (60%), and supporting reproducible builds (56%).

Although most developers followed many of the basic build best practices, more complex features, such as verifying provenance (36%) and cryptographically signing releases (attestation) (20%), struggled to garner significant adoption. Part of the problem may be a lack of common understanding of this terminology. In one of our interviews, a maintainer noted that even though many maintainers may not have heard of provenance or attestation, they do know how to describe the problem.

Figure 13 also shows that 24% of maintainers have no plans to use build services that cannot falsify metadata (provenance) or view provenance as not applicable. Similarly, 42% of maintainers and core contributors have no plans to have releases cryptographically signed or view this type of attestation as not applicable. These percentages are admittedly large, but there are instances where a maintainer is primarily maintaining a project for their own use, and trust is not an issue. Other users of such projects who have trust in the maintainer may not feel strongly about provenance or attestation. Finally, many OSS projects do not build the software directly at all; they only release source code. If an OSS project doesn't have a build service, then securing them and their results is irrelevant.

FIGURE 13

Does this project support the following build best practices?

2022 OpenSSF supply chain security survey, Q41, sample size = 72, DKNS excluded from the analysis

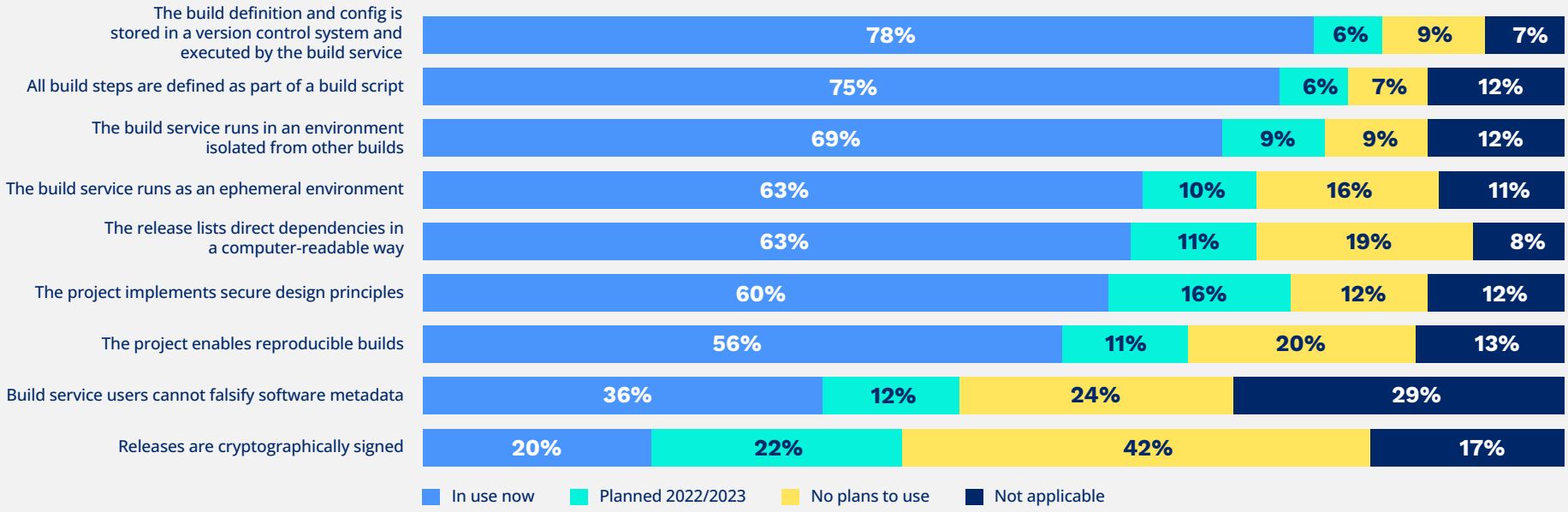
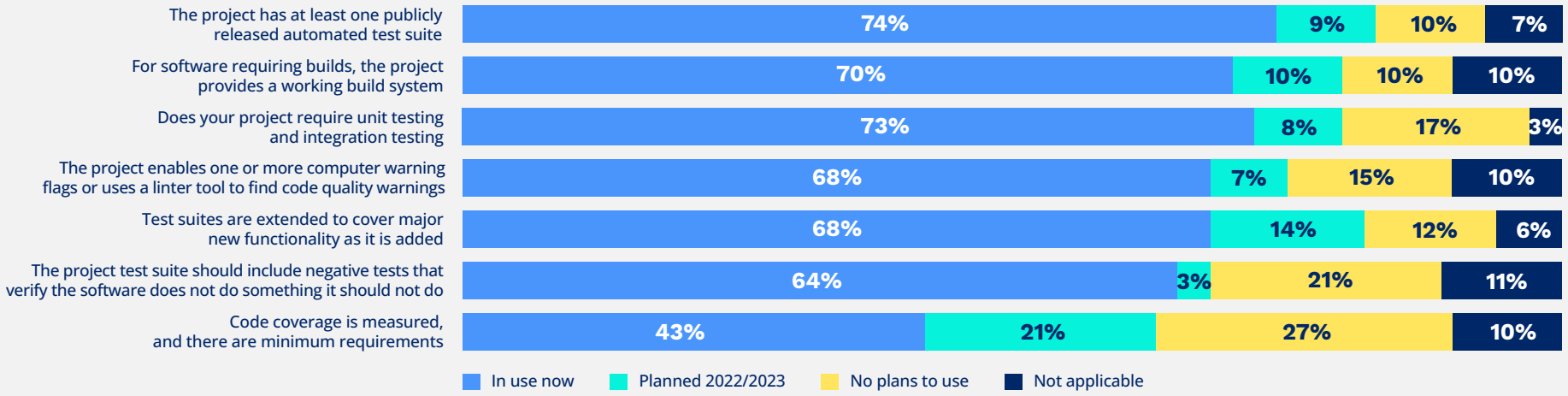


FIGURE 14

Does this project support the following quality best practices?

2022 OpenSSF supply chain security survey, Q42, sample size = 72, DKNS excluded from the analysis



Software quality best practices are well supported by maintainers and core contributors

Software quality has always been an important goal in some software development processes. As mentioned earlier, the sooner you can find a bug in the SDLC, the cheaper it is to resolve. Figure 14 shows the extent to which maintainers and core contributors engage in a selection of SQA best practices. The results are positive, especially for the more conventional approaches to SQA, including those shown in Figure 14. If we consider adoption to be a range from "in use now" to "in use now plus planned," adoption of the more conventional SQA best practices is very good for public test suites (74 to 83%), providing a working build system (70 to 80%), requirements for unit and integration testing (73 to 81%), compiler flags for code quality warnings (68 to 75%), and extending test suites as functionality expands (68 to 82%). This bodes well for efforts such as the **OpenSSF Compiler Options Hardening Guide for C and C++**, which provides extensive guidance when developing software in those languages.

The outliers in Figure 14 include negative tests (verifying that the software does not do something it should not do); many maintainers and core contributors have adopted (64%) this, but there are few plans to add them to projects that do not already do it. Minimum requirements for code coverage are modestly adopted, but with planned adoption reflecting 50% growth, this best practice is gaining traction. Code coverage measures the amount of code tested (e.g., the percentage of statements and / or percentage of branches), and it can provide quantitative evidence when the testing is poor (if many code statements or branches are completely untested, the test process will necessarily miss many problems).

Security in an area where more attention is required

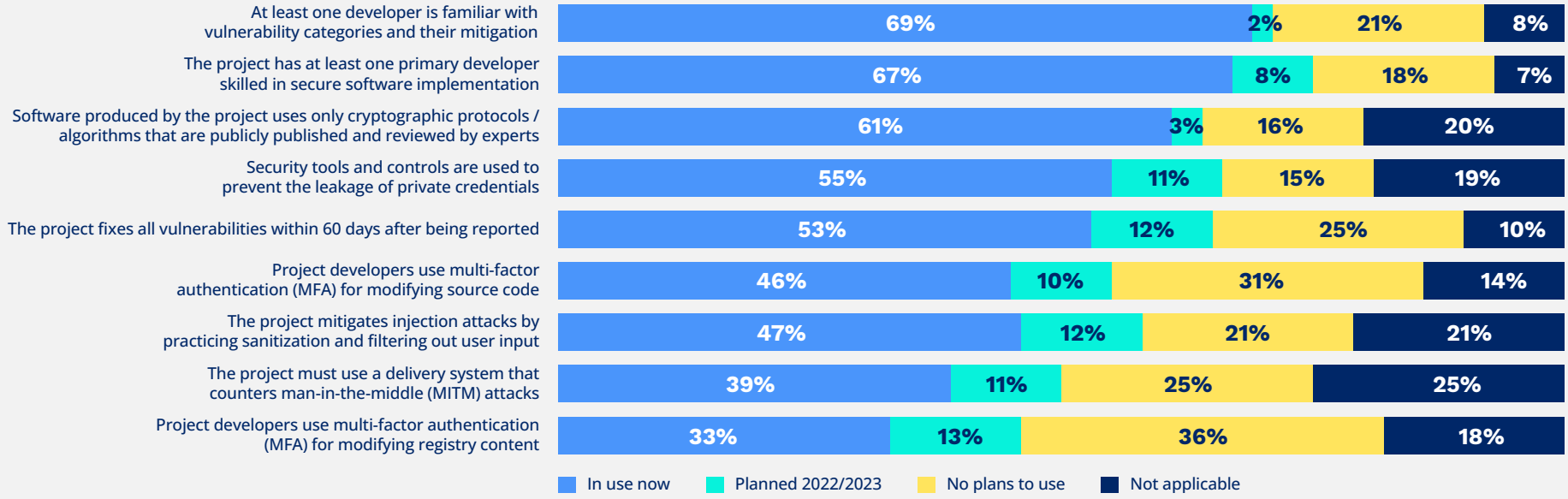
Security is an important element of software development these days. The attack on SolarWinds' Orion showed us that attackers can be quite sophisticated and brazen in their efforts to eavesdrop, control, or disrupt mission-critical systems, including the subversion of software build processes. For this reason, we looked at best practices to address the basic elements of software security in Figure 15.

Even considering the combination of maintainers and core contributors that have already adopted or are planning to adopt each of these security best practices, the results leave room for improvement. The highest threshold reached in Figure 15 was 75% for projects that have at least one developer skilled in secure software implementation. Today, most maintainers and core contributors barely adopt the use of security tools to prevent leakage of private credentials (55 to 66%) and fix vulnerabilities within 60 days after reporting (53 to 65%). However, in defense of maintainers, we can say that not all CVEs need fixing in a timely way, but the medium- or high-importance CVEs should and probably do get more immediate attention.

FIGURE 15

Does this project support the following security best practices?

2022 OpenSSF supply chain security survey, Q43, sample size = 72, DKNS excluded from the analysis



Best practices such as using multi-factor authentication (MFA) for modifying source code (46 to 56%) and using MFA when modifying registry content (33 to 46%) seem like areas where there is significant room for improvement. In the first part of 2022, various forges and package repositories began to move toward requiring MFA in certain cases. At the time, many expressed their excitement to see this, but some did resist it, perhaps seeing it as an unnecessary burden. The **OpenSSF even posted that it expressly supports movements toward MFA** and explained why this was becoming necessary to counter today's attackers. Since the time of this survey, MFA use has increased. We expect it will substantially increase further beginning in 2024 when **GitHub begins requiring 2FA to contribute code**.

Security tool demonstrates maintainer leanings toward manual reviews

We have already seen that OSS contributors often use SCA and SAST tools (Figure 9). At the same time, other OSS contributors have a stronger affinity for other security testing tools. Maintainers and core contributors rely more on manual code reviews. We attribute this emphasis on manual code reviews because of the need by maintainers and core contributors to gain a comprehensive understanding of how to modify, improve, and address cyber-security in their work.

Figure 16 confirms the focus that maintainers and core contributors have on resolving medium or high CVEs in a timely way (55 to 67%), the use of SCA tools (42 to 61%), and the use of SAST tools (41 to 51%). However, beyond the support that maintainers and core contributors derive from CSA and SAST tools, there is a limit to other tool use. We don't think this is necessarily an issue because investing the time to do manual code reviews supplemented by using key tools appears to us to be an acceptable approach to addressing this aspect of cybersecurity.

FIGURE 16

Does this project support the following security testing best practices?

2022 OpenSSF supply chain security survey, Q44, sample size = 72, DKNS excluded from the analysis

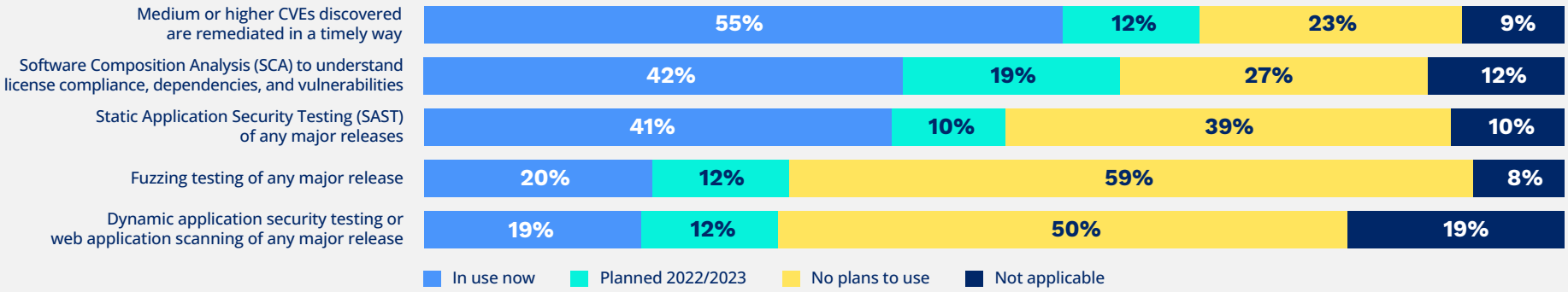
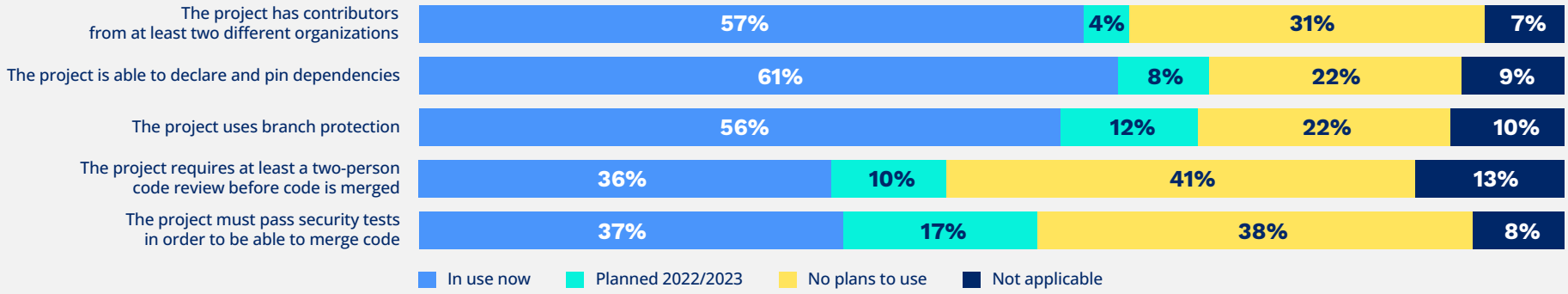


FIGURE 17

Does this project support the following secure source coding best practices?

2022 OpenSSF supply chain security survey, Q45, sample size = 72, DKNS excluded from the analysis



Support for secure coding principles varies by ease of administration

The Linux Foundation and OpenSSF have placed a significant emphasis on best practices for secure software development. Secure software development processes include secure software implementation, aka secure coding. While other processes are important when developing secure software (such as secure design and verification), it's vitally important to implement secure software. Figure 17 shows that maintainers and core contributors do largely embrace the selected secure coding practices identified, but there is still room for improvement.

Best practices that many maintainers and core contributors support include projects having contributors from two different organizations (57 to 61%), the ability to declare and pin dependencies (61 to 69%), and the use of branch protection (56 to 68%). Note that having "contributors from two different organizations" is not completely under the control of one organization; they can take steps to encourage it, but in the end, another organization must decide to join.

Secure coding practices that do not have a high level of support include requiring at least a two-person code review (36 to 46%) and having the project pass security tests before merging code (37 to 54%). The two-person code review is a worthy target, but its need is somewhat contingent upon the availability of others and the scope / complexity of the code being merged. A **2023 analysis by Josh Bressers** found that over half of all NPM releases have just one person maintaining them. A two-person code review can't happen when the project only has one person. Avoiding security testing before merging code appears to be a bad idea. However, project characteristics or constraints can often interfere. Not all maintainers or core contributors may have the skill and expertise in security testing to conduct security testing, the scale and scope of a project may not necessitate rigorous security testing, some projects rely on post-merge reviews instead of pre-merge reviews, continuous delivery needs can take precedence, and maintainer wisdom regarding what needs testing (and when) is important.

Open source contributor perspectives on how to improve software security and sustainability

Improving software security is an important and ongoing activity that is top of mind for the IT industry and governments. In this section of the report, we will look at OSS contributor perspectives on ways to improve the software supply chain, secure software development, and OSS sustainability.

Open source contributor guidance on improving open source software supply chain security

Figure 18 shows that the leading approaches to improving OSS supply chain security include making software security tools more intelligent (58%), increasing automation (54%), following best practices for secure software development (52%), and increasing OSS employer incentives (49%), security audits (47%), and peer review of source code (41%).

At 58% overall, OSS contributors widely use security tools, especially SCA and SAST tools. Other OSS contributors favor IaC and DAST tools. Security tools provide many benefits, including automatic detection of vulnerabilities, license compliance, continuous integration capabilities to prevent vulnerability introductions, better code quality, early detection and remediation of bugs, and making it easier for contributors to identify, prioritize, and resolve security issues.

More automation to eliminate pathways that could compromise security (54%) reduces time to market and developer fatigue and eliminates manual touch points, thereby reducing the available attack surface for bad actors. Automation is also a way to empower maintainers and core contributors without adding more burden to them. One respondent noted that more intelligent tooling and automation, as well as standardization, can solve the problem of transparency. Rather than making people change their workflow, providing security automatically or "by default" would be preferable. However, other respondents noted that there is only so much automated tools can do, and sometimes there are deeper problems to investigate. Moreover, there is a lot of work we need to do to make sure tools can speak to each other very well, such as integrating SBOM information and making it so easy to use a tool that users will always use it.

Following comprehensive best practices for secure software development (52%) confirms the value of having a trusted and published collection of best practices for addressing software security across the SDLC. The Linux Foundation and OpenSSF have already developed a comprehensive list of best practices for secure software development, and in the introduction of this report, we have identified where to find these best practices and the free training course on secure software development.

Increased incentives by employers (49%) can mean enabling employees to work on OSS projects important to the organization during working hours or providing additional incentives related to maintainership or community involvement. This is an important path to sustainability and one that helps organizations "give back" to OSS (often the OSS they vitally depend on) and is an important way to perpetuate / expand the OSS value proposition. Finally, respondents noted that employers and companies could play a big role in resourcing improvements for OSS. One respondent suggested having end-user enterprises that can benefit from open source fund maintainers. Since it seems that there is often no business case to pay maintainers for their support, OSPOs at companies and organizations can play a big role in thinking at a strategic level and supporting the OSS dependencies their organizations rely on.

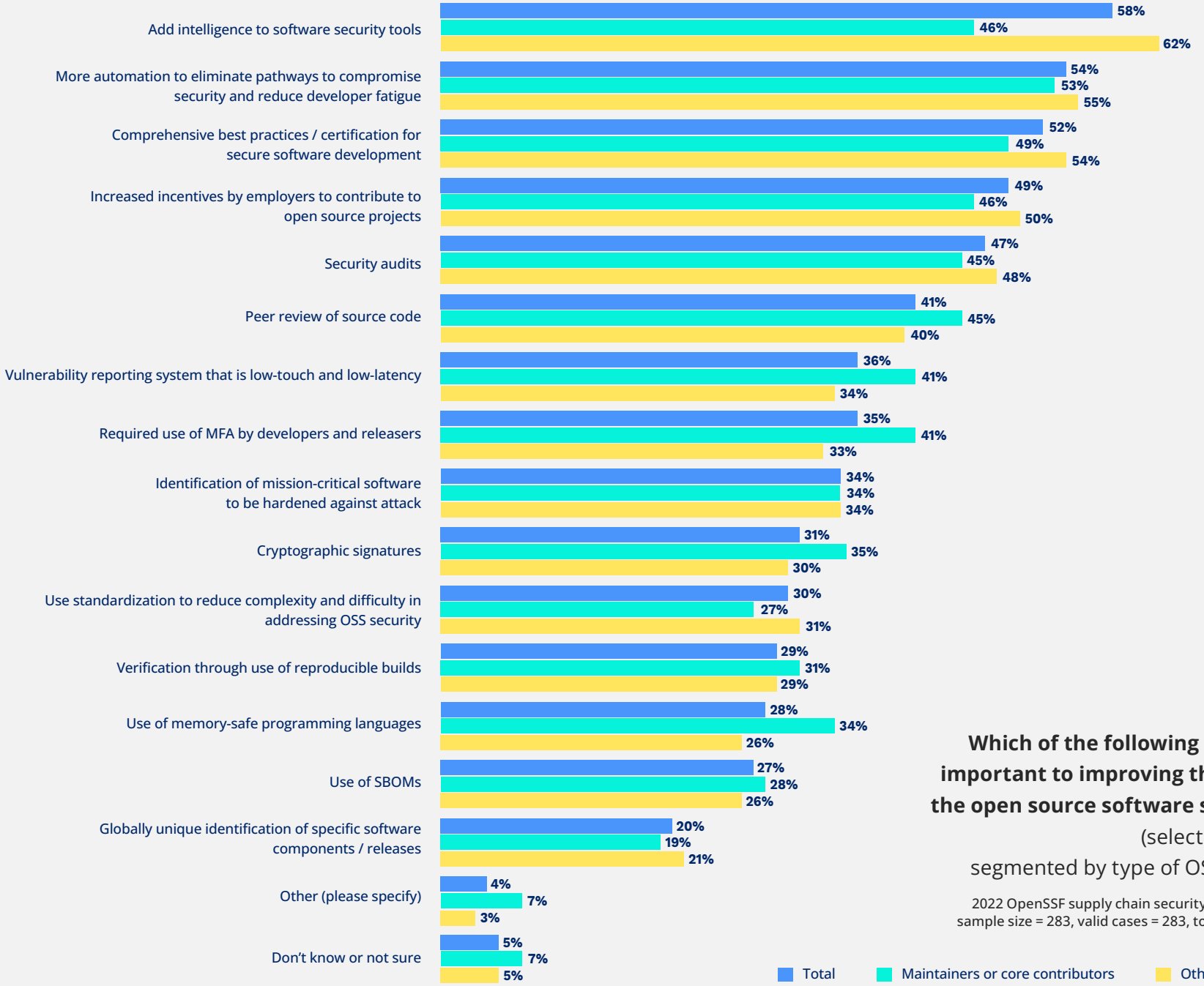


FIGURE 18

Which of the following activities are important to improving the security of the open source software supply chain?

(select all that apply)

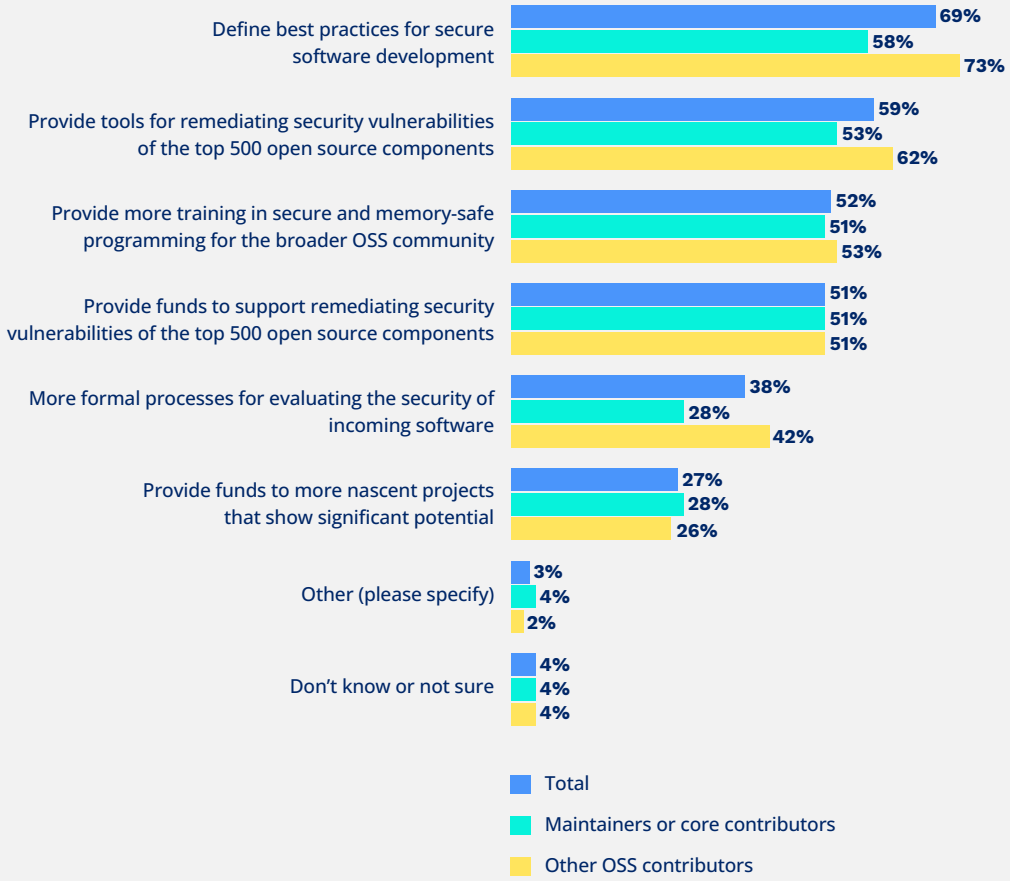
segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q48 x Q14a, sample size = 283, valid cases = 283, total mentions = 1,645

FIGURE 19

What are some of the ways that IT industry organizations could improve the security of developing open source software?
(select all that apply) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q50 x Q14a, sample size = 271, valid cases = 283, total mentions = 821



Based on past Linux Foundation research, security audits have often been addressed informally and sporadically, if at all. It is therefore exciting to see in Figure 18 that security audits (47%) are gaining visibility. They are a useful way to evaluate the security of key components as well as a key process for securing these components.

Finally, source code peer reviews (41%) are important to OSS contributors because taking the time to understand component functionality and security is the most effective way to ensure that code modifications, changes, and additions are performed carefully and securely, resulting in a high-quality component

Open source contributor guidance on improving open source development security

When we narrow the focus of the question to how to improve the development of OSS across the supply chain, we see some overlapping trends and some additional findings. Figure 19 shows that the leading guidance from OSS contributors is to define best practices for secure software development (69%). The best practice definitions for secure development and training courses appear to be resources that many OSS contributors are not aware of. See the introduction of this report for links to both of these resources. Many respondents also noted that IT organizations could also work on defining best practices; however, some respondents expressed concerns that guidelines developed by enterprises may get too long or cumbersome, and maintainers may not read or implement them. Another respondent noted that it would be useful to get bigger organizations together to talk about what has worked and what hasn't when it comes to securing their projects.

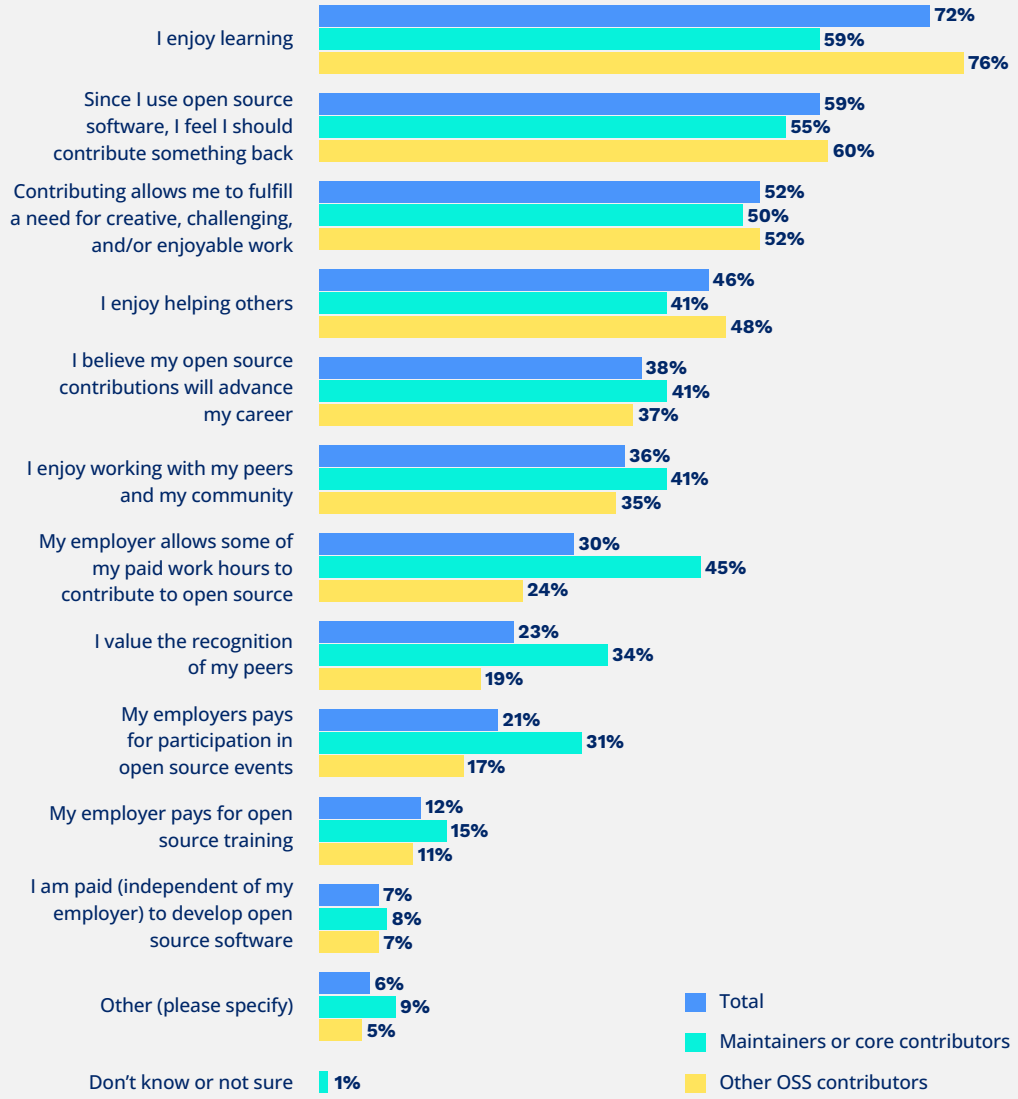
A second tranche of responses in Figure 19 includes providing tools to remediate vulnerabilities in the top 500 OSS components (59%), more training in secure and memory-safe programming

FIGURE 20

What drives your motivation for maintaining and contributing to OSS projects and components?

(select all that apply) segmented by type of OSS contributor

2022 OpenSSF supply chain security survey, Q51 x Q14a, sample size = 271, valid cases = 283, total mentions = 1,088



(52%), and funding to remediate vulnerabilities in the top 500 OSS components (51%).

Project Alpha-Omega is an initiative within the OpenSSF aimed at identifying and securing the most critical open source projects. The alpha part of the project is to proactively secure the most widely used and critical open source projects. See **Alpha-Omega—Open Source Security Foundation (openssf.org)** for more information on this project.

More training in memory-safe programming languages is also a great way to improve OSS security. Rust (ownership model) and Swift (automatic reference counting) take rigorous approaches to memory management and are "safer" languages based on their memory management techniques as compared to C or C++, while still providing good performance. Most popular languages, including Go, Java, JavaScript, C#, and Python, provide memory safety in part by including automatic garbage collection; this abstracts away much of the complexity of memory management to make it easier for developers. The choice of language depends on the specific project requirements such as performance needs, ecosystem considerations, and developer expertise. Selecting a memory-safe programming language automatically prevents many vulnerabilities that plague those developing in languages such as C and C++.

Open source contributor guidance on improving open source sustainability

Figure 20 shows that personal and altruistic beliefs drive OSS contributors. On a personal level, OSS contributors enjoy learning (72%). Contributing provides them with a way to fulfill creative, challenge, and enjoyment needs (52%). On an altruistic level, OSS contributors use OSS and feel they should contribute something back (59%) and believe OSS involvement is a way to help others (46%).

Maintainers and core contributors also emphasize (relative to other OSS contributors) that OSS contributions will advance their careers (41%). They find enjoyment in working with their peers and the OSS community (41%), and employers often do allow work on OSS during working hours (45%).

Solving the problem of open source sustainability is complex and requires a multi-dimensional approach. Since altruistic motives often drive open source projects, balancing the need for ongoing maintenance and development with limited resources can be challenging. Corporate stewardship and partnerships, as well as funding and financial support by employers, are a new and welcome phenomenon, although they cannot be at the expense of compromising the satisfaction that OSS contributors gain from their involvement in OSS and with the community. Support for community engagement and growth is an important objective so that a path exists from occasional OSS contributor to maintainer. Building a healthy community culture with recognition, rewards, and an established code of conduct ensures community participation is a supportive experience. Finally, obtaining a balance between addressing these personal, organizational, and cultural needs while continuing to evolve and advance the process model and improving OSS security and quality in a way that encourages rather than discourages involvement is absolutely paramount.

Conclusions

As we work on efforts to improve the security of the OSS ecosystem, we should empower OSS maintainers to build security into their products and build processes, automating rote tasks and reducing the effort needed by maintainers. We hope this survey has helped shed light on what practices have worked and where we still need to go in the future. Key conclusions based on the data analyzed in this report are as follows.

Keep the open source maintainer perspective in mind when thinking about open source sustainability

Software development is a human endeavor. Even when AI / ML is used to help generate software, humans must define what is to be done, as well as review and fix the results. Software development is also about problem-solving, where the developer or maintainer must solve a functional problem that is often bounded by constraints, such as security. To solve problems, an OSS contributor (be they a maintainer, core contributor, occasional contributor, or one-time contributor) should understand the existing code to some extent before making any decisions and taking action. As projects grow in size, understanding the code becomes more difficult because of the code's increased size and complexity. Manual code reviews are therefore an important element of what maintainers and core contributors do. A positive finding from this survey was the high priority that OSS maintainers and core contributors attach to code reviews and peer reviews. The importance of these activities to maintainers and core contributors was greater than some might have expected. Another way to say this is that any efforts to improve security must include these voices in the room; security work must be a collective effort and prioritize collaboration.

OSS cybersecurity is an important issue that the OSS community must successfully address. Gaps or lapses in security that lead to embarrassment, financial harm, or unauthorized data exposures can do more than just give OSS a black eye. Past research shows that OSS's image benefits from its community-driven development approach, which provides an opportunity for the security of OSS components to be better than proprietary code. The OSS community must live up to this expectation and find ways to address security needs without doing irreparable harm to the community culture.

The use of security tooling and automation is key to addressing open source cybersecurity

Throughout this report, security tool use by OSS contributors has been an important ingredient in addressing cybersecurity needs. SAST and SCA tools have found favor with many OSS contributors, but some OSS contributors are open to also using other security tools such as DAST and IaC. The use of all four of these functional tool categories should provide an excellent foundation from which to address security testing and automation.

Tools can reduce the burden on individual developers and help achieve ecosystem-wide success. Tooling can improve both transparency and security, but it should not be a burden on maintainers; rather, tooling should work with developers' existing workflows and provide security features by default. We also need more investments in interoperability to make different tools work together, and we must be realistic and understand that tooling and automation cannot solve all problems.

Education and best practices are an important part of the solution

A significant portion of this report highlights the best practices that maintainers and core contributors are following. Figures 11 to 17 showcase maintainer use and planned implementation of best practices for secure software development and indicate that some best practices have widespread adoption. Many others are likely to have widespread adoption by the end of 2023. While maintainers and core contributors are not strangers to best practices, when we asked about how to improve OSS supply chain security (Figure 18) and the security of software development (Figure 19), support for best practices was always a leading response. This suggests that there may be a knowledge gap with many OSS contributors not being aware that the Linux Foundation has already defined a comprehensive list of best practices for secure software development across the SDLC as well as a training course in secure software development. For links to these resources, see the introduction in this report. This suggests a need for more marketing to help them become aware of them.

As Robert Scholte noted, "You don't know what you don't know." More broadly, there are tools and improvements that users and maintainers can use, but they need to be aware of them and their use first. The significant number of "Don't know or not sure" (DKNS) responses to our surveys also indicate that education about the security tools available is a good first step. Finally, according to Brian Demers, the current undergraduate level of education for software development—in computer science (CS), software engineering (SwE), and similar—does not touch on security, which is a big problem. The basics of how to develop secure software should be part of every undergraduate curriculum for CS, SwE, and similar fields.

Methodology

About this study

A web survey conducted by Linux Foundation Research and its partners in March 2022 served as the basis for this study. The survey's goal was to provide a global perspective on the state of open source supply chain security. Published in June 2022, an initial report, Addressing Cybersecurity Challenges in Open Source Software, presented some of the key findings from the survey. However, this initial report did not include the entire section of the survey that collected data from maintainers and core contributors on the adoption of best practices for secure software development because of resource, length, and time constraints. This additional report, Maintainer Perspectives on Open Source Software Security, published in December 2023, provides this best practice data and includes additional context based on a segmentation of OSS contributors. The two reports do not overlap except for the demographic data.

In this section, we present the study methodology and the demographics of the respondents. From a research perspective, it was important to counter sample bias and ensure high data quality. We handled the elimination of sample bias by sourcing our usable sample from the Linux Foundation membership, partner communities, social media, and a third-party panel provider. We addressed data quality through extensive pre-screening, screening criteria, and data quality checks to ensure that respondents had sufficient open source familiarity and professional experience to answer questions accurately on behalf of the organization they worked for.

The Open Source Supply Chain Security Survey comprised 55 questions and focused on the following:

- General software security
- OSS software security (includes questions on the adoption of best practices for secure software development)
- How to improve OSS software security

We collected survey data from end-user organizations, IT vendors and service providers, and nonprofit, academic, or government organizations. Respondents spanned many vertical industries and companies of all sizes, and we collected data from geographies that included the Americas, Europe, and Asia Pacific.

We conducted the 2022 Open Source Supply Chain Security Survey in March 2022. While this data is aging, it still provides useful insights into the challenges that maintainers face and the decisions that they make regarding how to address secure software development. The derivation of the sample size discussed in this report is as follows:

1. This survey had 1,175 respondents who started the survey.
2. Screening criteria (Q1-3,5,6) removed 437 respondents, leaving 738.
3. Of these 738, only 539 completed the demographic questions and began answering supply chain security questions.
4. Of these 539, 441 self-identified as either an open source maintainer, core contributor, occasional contributor, one-time contributor, or other contributor.
5. Of these 441, just 159 respondents self-identified as either an open source maintainer or core contributor, and 282 were other OSS contributors (occasional, one-time, or other contributors)

6. Of these 159, only 72 elected to answer specific questions regarding how they addressed secure software development.

Although there was a requirement for respondents to answer nearly all questions in the survey, there were times when the respondent was unable to answer a question because it was outside the scope of their role or experience. For this reason, we added a DKNS response to the list of responses for nearly all questions. However, this creates a challenge regarding how we should interpret the DKNS responses.

One approach is to treat a DKNS just like any other response so that we know the percentage of respondents that answered DKNS to a question. The advantage of this approach is that it reports the exact distribution of data collected. The challenge with this approach is that it can distort the distribution of valid responses, i.e., responses collected where respondents could answer the question.

Some of the analyses in this report exclude DKNS responses. This is a decision made where (a) it is important to understand the distribution of responses excluding DKNS and (b) we can classify the DKNS data as either missing at random or missing completely at random. Excluding DKNS data from a question does not change the distribution of counts for the other responses, but it does change the size of the denominator used to calculate the percent of responses across the remaining responses. This has the effect of proportionally increasing the percentage values of the valid responses. Where we have elected to exclude DKNS data, the footnote for the figure includes the phrase "DKNS responses excluded."

The percentage values in this report may not total exactly 100% due to rounding.

Data.World access

Linux Foundation Research makes each of its empirical project datasets available on Data.World. Included in this project dataset are the survey instrument, raw survey data, screening and filtering criteria, and frequency charts for each question in the survey. You can find Linux Foundation research datasets, including this project (2022 Open Source Software Supply Chain Security Survey) at **data.world/thelinuxfoundation**.

Acknowledgments

We thank all the participants of the survey for sharing their insights and experience on the state of secure software development. Special thanks to peer reviewers, Linux Foundation colleagues, and an array of maintainers for their perspectives and involvement in the various stages of this research: Omkhar Arasaratnam, Stephen Augustus, Brian Behlendorf, Hilary Carter, Brian Demers, Adrienn Lawson, Kim Lewandowski, Oleg Nenashev, Nick O'Leary, Jed Salazar, Robert Scholte, Daniel Stenberg, Kate Stewart, Liran Tal, Harry Toor, Dana Wang, David A. Wheeler, and Adolfo Garcia Vettia.

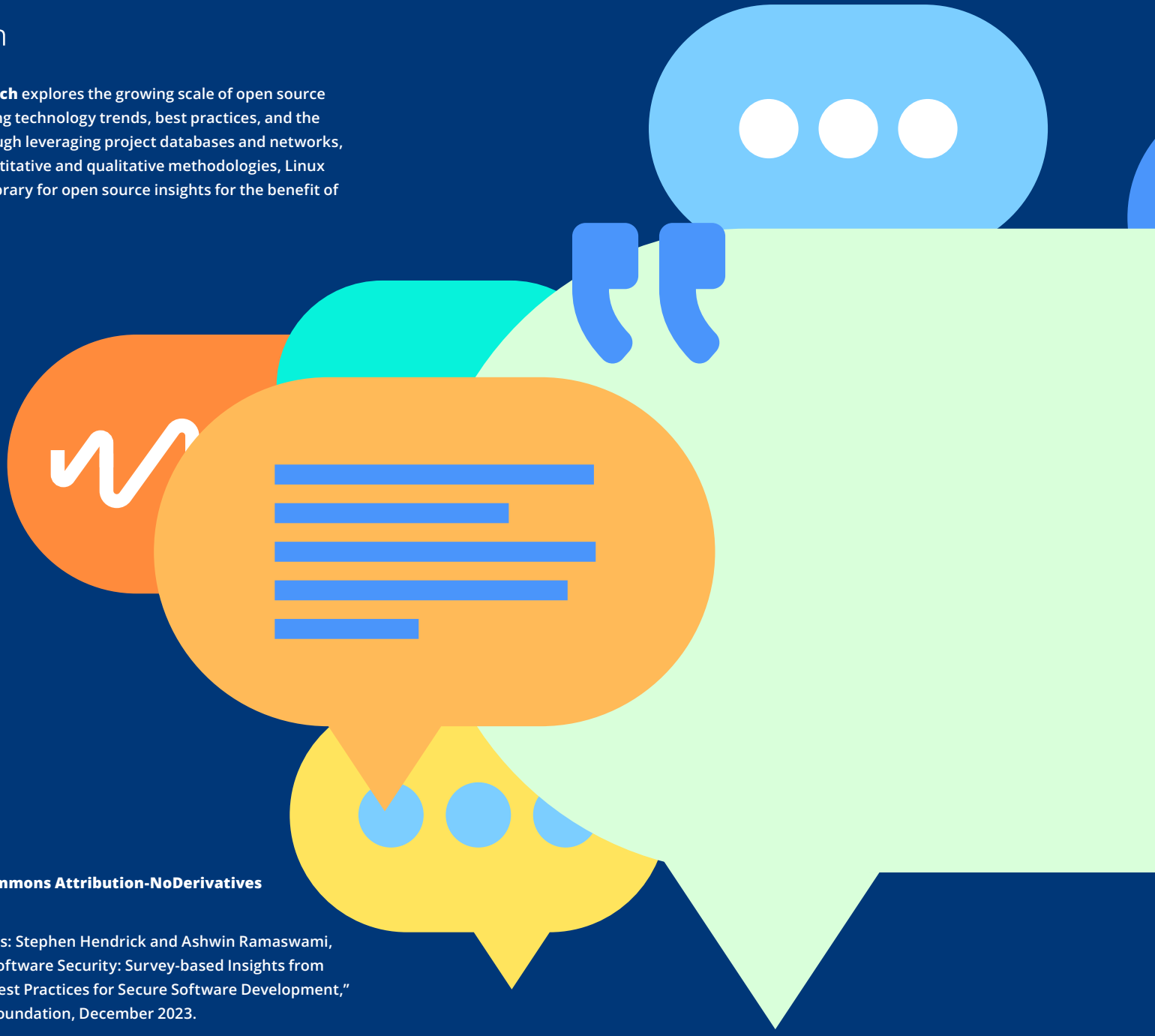
About the authors

STEPHEN HENDRICK is vice president of research at the Linux Foundation, where he is the principal investigator on a variety of research projects core to the Linux Foundation's understanding of how OSS is an engine of innovation for producers and consumers of IT. Steve specializes in primary research techniques developed over 30 years as a software industry analyst. Steve is a subject matter expert in application development and deployment topics, including DevOps, application management, and decision analytics. Steve brings experience in a variety of quantitative and qualitative research techniques that enable deep insight into market dynamics and has pioneered research across many application development and deployment domains. Steve has authored over 1,000 publications and provided market guidance through syndicated research and custom consulting to the world's leading software vendors and high-profile start-ups.

ASHWIN RAMASWAMI is an open source maintainer, developer, and policy researcher. He also works on web application architecture and cybersecurity and research and writing with the Linux Foundation. He holds a B.S. in computer science from Stanford University and is pursuing a J.D. degree at Georgetown Law.

THE LINUX FOUNDATION | Research

Founded in 2021, **Linux Foundation Research** explores the growing scale of open source collaboration, providing insight into emerging technology trends, best practices, and the global impact of open source projects. Through leveraging project databases and networks, and a commitment to best practices in quantitative and qualitative methodologies, Linux Foundation Research is creating the go-to library for open source insights for the benefit of organizations the world over.



To reference this work, please cite as follows: Stephen Hendrick and Ashwin Ramaswami, "Maintainer Perspectives on Open Source Software Security: Survey-based Insights from Maintainers Regarding How They Address Best Practices for Secure Software Development," foreword by Stephen Augustus, The Linux Foundation, December 2023.