THE 2024 TIDELIFT STATE OF THE OPEN SOURCE MAINTAINER REPORT

September 2024

Top headlines from our annual survey of open source maintainers

TIDELIFT

Introduction

In mid-2024, Tidelift fielded its third survey of open source maintainers. Over the following pages, we'll share twelve of the most compelling headlines with you.

More than 400 maintainers responded and shared details about their work, including how they fund it, who pays for it, and what kinds of security, maintenance, and documentation practices they have in place today or would consider in the future. They also shared their thoughts about some "in the headlines" issues like the recent xz utils hack and the impact of AI-based coding tools.

ABOUT THE SURVEY

This is the sixth year in a row Tidelift has conducted a survey about open source and the third time it focused exclusively on the maintainers who create and maintain the open source projects we all depend on. We contacted maintainers through a social campaign in July and August 2024. After screening for quality and completeness, we analyzed the answers from 437 respondents who maintain at least one open source project.

HEADLINES

HEADLINE #1

60% of maintainers are (still) not paid for their work

In our 2023 state of the open source maintainer report, we asked maintainers to describe whether they consider themselves to be an unpaid hobbyist or a paid professional maintainer. We gave them four choices:

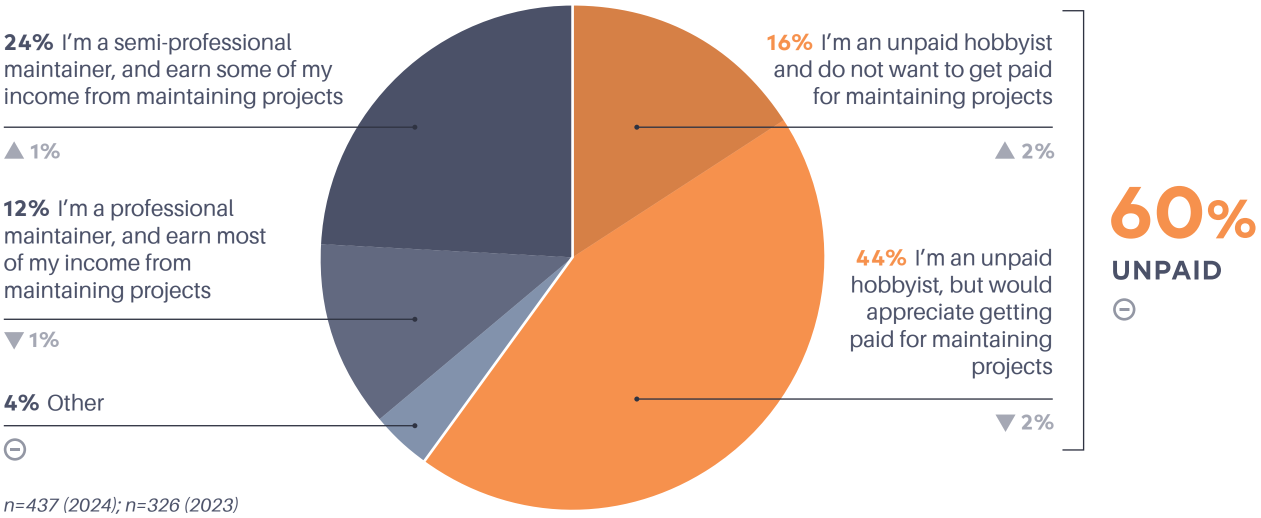
1. I'm an unpaid hobbyist and do not want to get paid for maintaining projects
2. I'm an unpaid hobbyist, but would appreciate getting paid for maintaining projects
3. I'm a semi-professional maintainer, and earn some of my income from maintaining projects
4. I'm a professional maintainer, and earn most or all of my income from maintaining projects

The most cited stat from that previous survey was that 60% of maintainers described themselves as unpaid hobbyists. We asked the same question again this year to see if things had changed. As it turns out, they have not changed a bit.

60% of maintainers are (still) not paid for their work

Which of the following phrases best describes how you approach your role as an open source maintainer?

▲▼ % point change from last year. ⊖ = no change



n=437 (2024); n=326 (2023)

As you can see in the chart on the previous page, even with a larger sample of maintainers filling out this year's survey, the percentage of maintainers who describe themselves as unpaid hobbyists stayed identical: 60%. Sixteen percent of maintainers said they were unpaid hobbyists and would not want to get paid (compared to 14% in 2023), and 44% said they were unpaid hobbyists but would appreciate getting paid (compared to 46% in 2023).

Meanwhile, the percentage of maintainers saying they earn most or all of their income from maintaining projects is almost identical at 12% this year versus 13% in 2023. And the percentage of semi-professional maintainers was 24% this year and 23% in 2023.

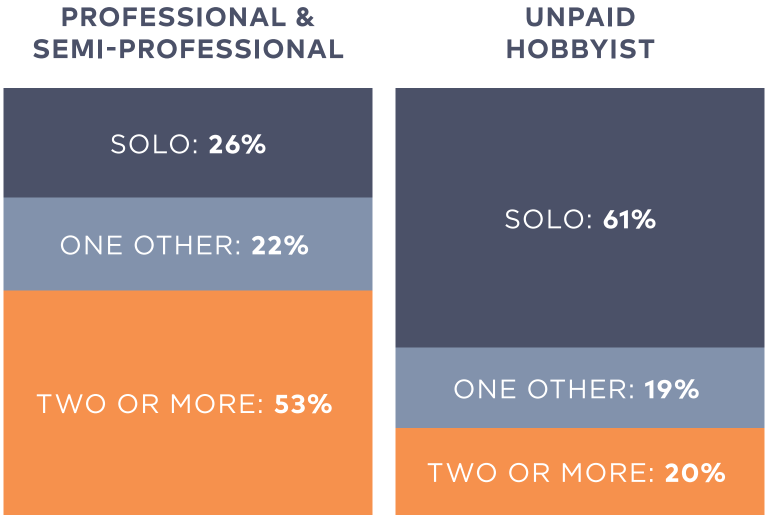
Full disclosure: it would have been awesome if this headline was different, if we'd found that the percentage of maintainers being paid for their work had increased significantly over the past year. But the fact is that things haven't changed, and especially in the year of the xz utils hack and with increased focus by both governments and organizations on the importance of the secure software supply chain, this is a newsworthy—and disappointing—finding to report.

Are paid maintainers more likely to have co-maintainers?

We were interested in finding out if there is any correlation between being a paid (professional or semi-professional) maintainer and the number of co-maintainers a project has, and it turns out there is.

Paid maintainers are more likely to have co-maintainers, unpaid maintainers are more likely to be flying solo

How many people maintain the project or projects you maintain?



Professional maintainers, n=42; Semi-professional maintainers, n=91; Unpaid hobbyist maintainers, n=227. Totals do not equal 100% because of rounding.

Over half of maintainers (53%) who describe themselves as paid maintainers have two or more co-maintainers on their projects. Only 26% of this group is made up of solo maintainers.

Meanwhile the opposite is true of unpaid maintainers. Sixty-one percent of unpaid maintainers are solo maintainers, with only 20% of unpaid maintainers having more than two co-maintainers.

What do we make out of this? It's hard to definitively say what is a cause and what is an effect here. Are projects with more maintainers simply larger projects that are able to command more income? Or because their maintainers are getting paid for their work, they are able to entice more people to help? Similarly, perhaps unpaid maintainers are unpaid because their projects are relatively new or haven't attracted a ton of interest? Or maybe they are unable to bring in more co-maintainers because there isn't money to fund the work?

Interestingly, in one example of how this particular finding impacts project health and security, the OpenSSF SLSA authors (SLSA is a set of standards and technical controls that can be adopted to improve project integrity) believe that having multiple maintainer projects is a best practice. But they had to remove mandatory two-person review of all changes from version 1.0 until this solo maintainer issue is addressed (the OpenSSF Scorecard does still recommend a two-person review when feasible as a security best practice).

In later findings, we'll delve into some additional data about paid and unpaid maintainers, including exploring differences in the security and maintenance practices paid maintainers are able to implement versus their unpaid counterparts. But because this has been the most often quoted statistic from our previous maintainer survey, we wanted to update the "60% of maintainers are not paid for their work" headline.

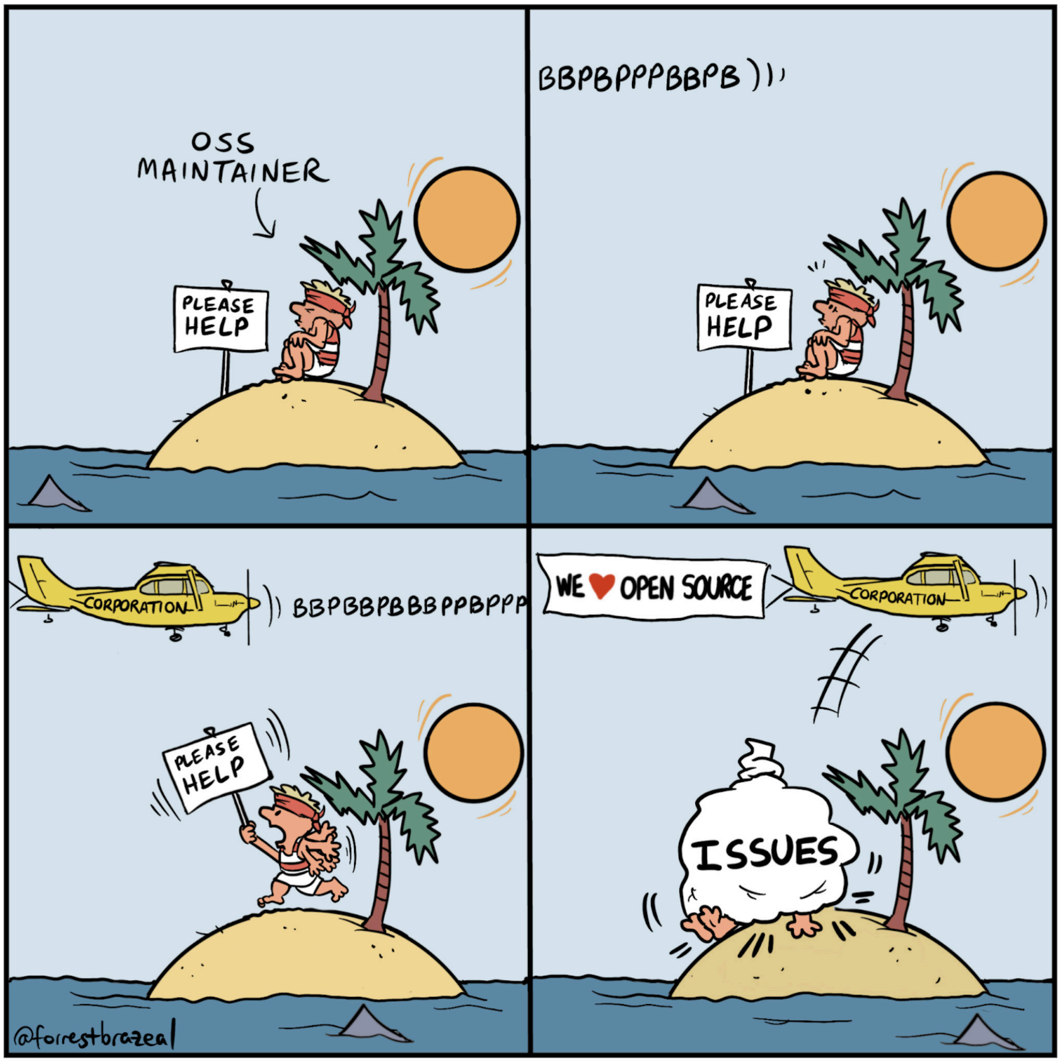
Unfortunately, the update is that it has not changed. ■

HEADLINE #2

The more maintainers are paid, the more improvements they make to their projects

In most corners of humanity, it is understood that people are typically willing to do more work *if you pay them* than they will do for free. Yet the open source world can sometimes feel like an upside down universe where there is an expectation that unpaid or underpaid volunteers will maintain their projects to the same standards that an organization would expect from their own employees, who often get paid handsomely.

For open source maintainers this can be ... frustrating, as Forrest Brazeal captures in this cartoon.



What kinds of improvements are paid maintainers able to make?

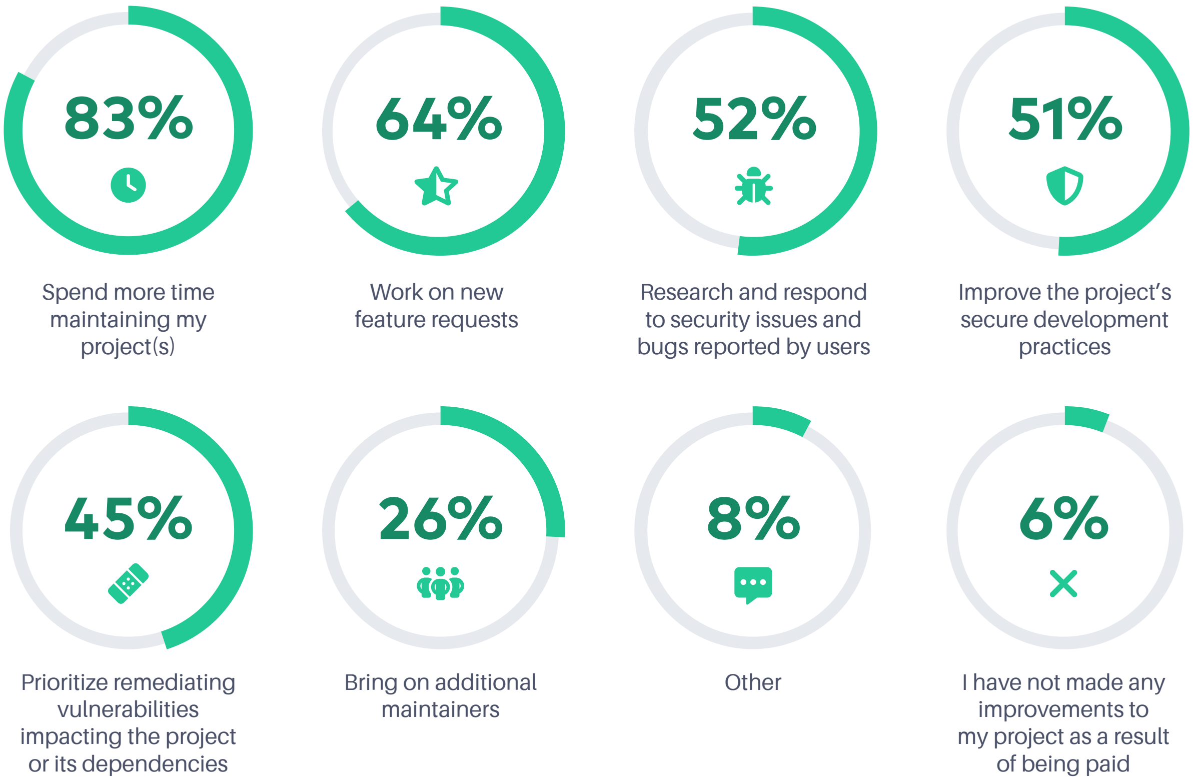
While it might not be rocket science to reach the conclusion that open source maintainers would do more work if they were paid than they can do for free, we wanted to use this year's survey to get some additional data points to support that conclusion.

In the previous finding, we reported that 60% of maintainers describe themselves as unpaid hobbyists, and 36% of maintainers describe themselves as paid (professional or semi-professional) maintainers, earning some or all of their income from their open source work.

We asked the 36% of maintainers who are getting paid for their work what types of improvements they've been able to make to their projects as a result of getting paid. The vast majority of paid maintainers (83%) report that they can spend more time maintaining their projects as a result of being paid. No shock there. But what else can they do when they are being paid?

Improvements made by paid maintainers

Paid maintainers were asked, "Which of the following improvements have been made to your projects(s) as a result of getting paid for your work as an open source maintainer? (Choose all that apply)



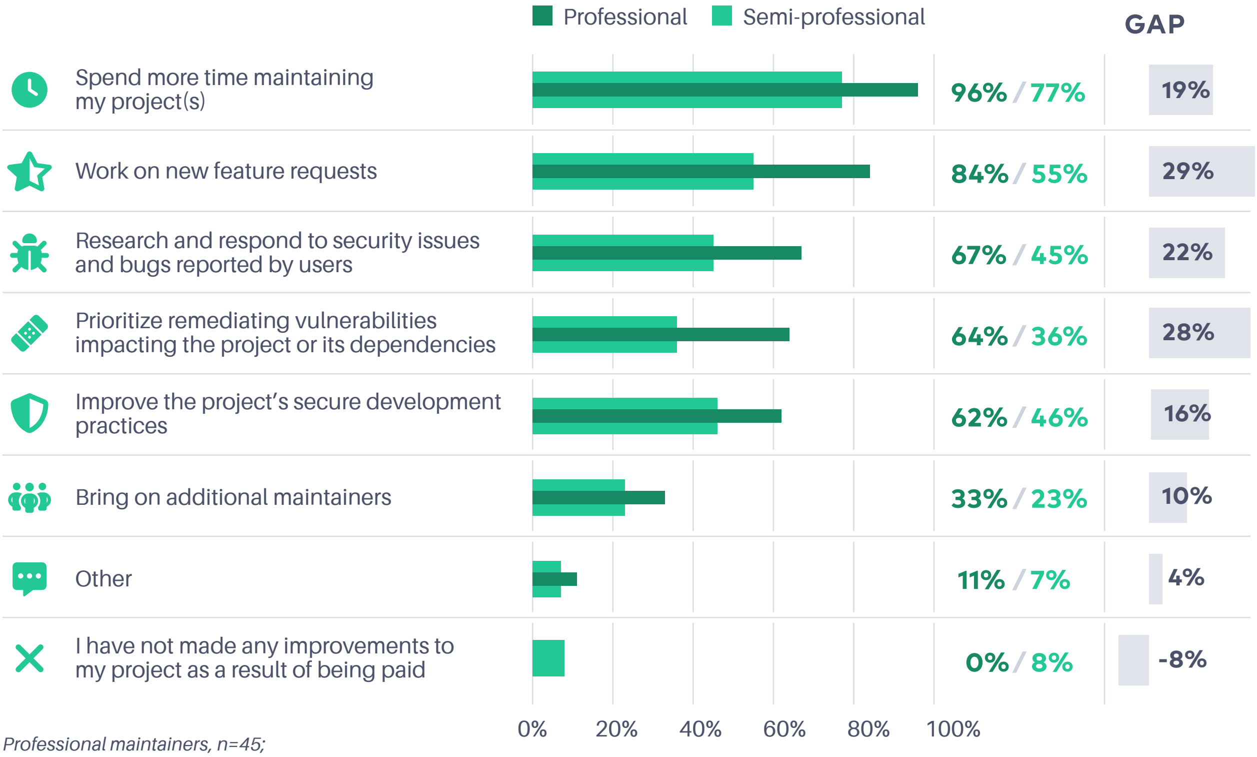
Sixty-four percent of paid maintainers report that they can work on new feature requests, 52% are better able to research and respond to security issues and bugs reported by users, 51% can improve their project's secure development practices, and 45% can prioritize remediating vulnerabilities impacting the project or its dependencies.

When you break down the paid maintainers into professional (earning most or all of their income from their maintenance work) and semi-professional (earning some of their income from maintaining projects), it becomes clear that the amount of money a maintainer is making for their work has a large impact on the types of improvements they are able to make. Across nearly all major categories, professional maintainers are on average **over 20 percentage points more likely to make key improvements** to their projects than semi-professional maintainers.

(Side note: throughout this report, you'll see us show gaps between percentages, like in the chart below, consistently. It's our feeling that the brain processes these gaps more easily than it does a percentage difference between two numbers. For example, if one number was 50% and the next one was 75%, the gap between those two percentages would be 25%, but the percentage difference between the two is actually 50%. So for readability you'll see us refer to percentage points gaps in most places although occasionally when the differences are really compelling you'll see us refer to a percentage difference between two numbers or even something like 3x or 3 times another number.)

Professional maintainers are 20-30 percentage points more likely to make key improvements to their projects

Paid maintainers were asked, "Which of the following improvements have been made to your project(s) as a result of getting paid for your work as an open source maintainer? (Choose all that apply)."



Professional maintainers, n=45;
Semi-professional maintainers, n=97

More detail: almost *all* professional maintainers (96%) can spend more time maintaining their projects (as compared to 77% of semi-pro maintainers) because they are getting paid. Eighty-four percent of professional maintainers can work on new feature requests, as opposed to 55% of semi-pro maintainers. And perhaps most importantly, professional maintainers are **almost twice as likely (64%) to be able to prioritize remediating security vulnerabilities** impacting their project or dependencies compared to semi-pro maintainers (36%).

Maintainers who get paid (still) spend more time working on their projects

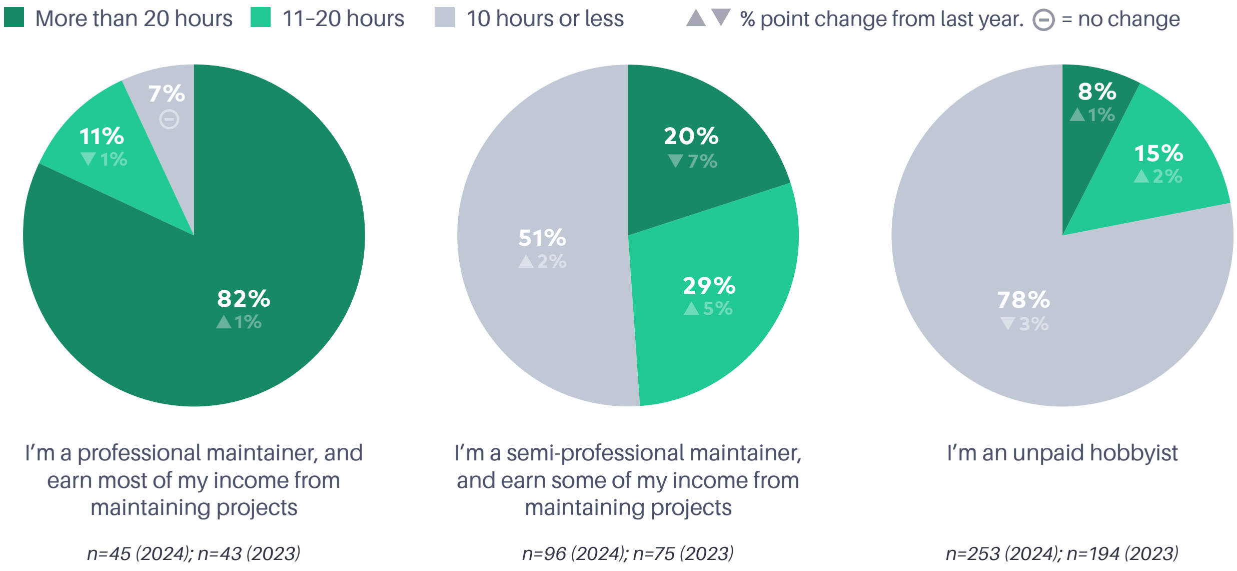
In last year's report, we shared data showing that the more maintainers get paid, the more they work on open source. This conclusion still holds true in the 2024 data, and the results were remarkably consistent.

In the previous study, 81% percent of professional maintainers earning most or all of their income from maintaining projects spend more than 20 hours a week maintaining their projects. This year, the percentage was nearly identical (82%).

Conversely, in last year's survey, we found that the vast majority of unpaid hobbyists spend *ten hours or less* per week on their maintenance work (81%). This percentage also stayed consistent in this year's survey, with 78% of unpaid hobbyist maintainers working ten hours or less per week.

The more maintainers get paid, the more they work on open source

How much time do you spend per week maintaining open source projects?



I'm a professional maintainer, and earn most of my income from maintaining projects

n=45 (2024); n=43 (2023)

I'm a semi-professional maintainer, and earn some of my income from maintaining projects

n=96 (2024); n=75 (2023)

I'm an unpaid hobbyist

n=253 (2024); n=194 (2023)

Totals do not equal 100% because of rounding.

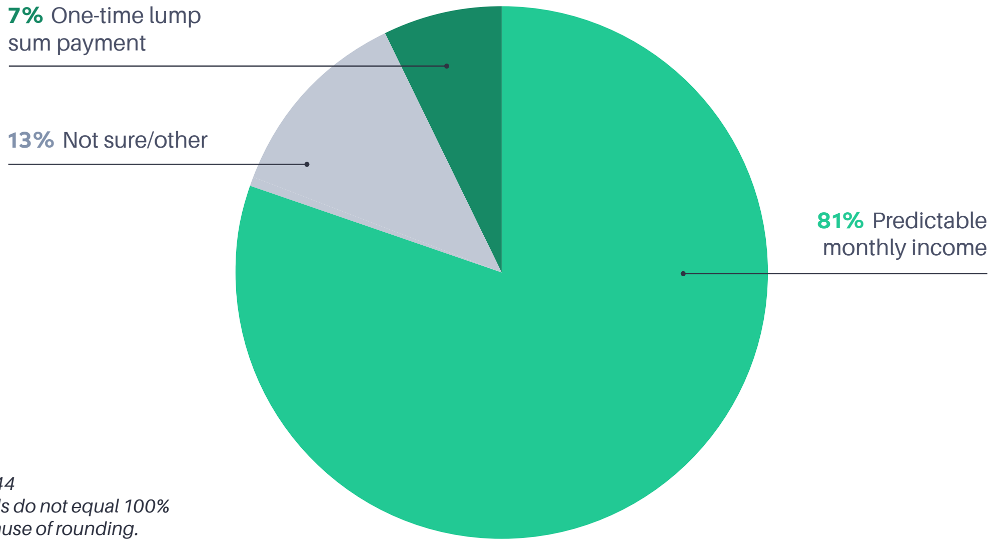
How do maintainers like to get paid?

We’ve heard from many maintainers that how they are paid for their work also matters. For many maintainers there is a huge difference between getting a one-time “airdrop” of money, perhaps right after a high profile incident where people are paying attention to their projects, compared to ongoing recurring income that they can count on. So this year for the first time we asked maintainers to tell us whether they would prefer to get predictable monthly income or a one-time lump payment.

An overwhelming majority of maintainers prefer to receive predictable monthly income, with 81% choosing that option.

Maintainers prefer predictable, recurring income for their projects

When it comes to ensuring you can complete ongoing security and maintenance work for your project(s), would you prefer to receive predictable monthly income or a one-time lump sum payment?



n=344
Totals do not equal 100% because of rounding.

Gary Gregory, a co-maintainer of Apache Commons and the high-profile Log4j package among other important Java projects, has a strong perspective on the issue of one-time payments versus recurring income. When talking about the difference between the income Tidelift provides and one-time project grants like the Log4j team received after the Log4Shell incident, he said:

> “I think the recurring income piece is critical for me, at least. Just imagine what it’s like to have a job with a recurring income—it makes you feel safe, secure, and confident that you can keep on doing this work and that it’s not time wasted. It also lets you plan ahead. I always maintain a list of the components I want to release in the near future, and then I have a longer-term list of things that I want to work on, that I know I’ll get to.”

Again, you do not need a PhD in economics to understand that when people are paid, they will do more than when they are not paid, and that the more you pay them, the more they are willing to do. But this year’s survey gives us a few different lenses through which to explore the improvements organizations can expect to see when they prioritize paying the maintainers of the projects they use. If having healthy, well-maintained, and secure open source dependencies is a priority for your organization, ensuring your maintainers themselves are financially healthy and well-maintained should be a priority, too. ■

HEADLINE #3

Who’s paying the maintainers? Donation programs, employers, and Tidelift

In each of our maintainer surveys over the last few years, we’ve asked a variety of questions to learn more about how maintainers get paid for their work. In our first finding from this year’s report, we shared the results from a question where we forced maintainers to categorize themselves as a professional, semi-professional, or unpaid hobbyist, and 60% of maintainers placed themselves in the unpaid hobbyist category (they could only choose one answer).

Later in the survey we asked a slightly different question about maintainer income, and allowed them to choose as many answers as applied to them:

> Which of the following describe the source of your maintainer income (Choose all that apply)?
>
> - Salary or wages from an employer because maintenance is an explicit part of my job responsibilities
> - I receive income from another organization or individual (e.g. Tidelift, GitHub Sponsors, foundations, etc.) to maintain a project
> - I don’t get paid to maintain projects

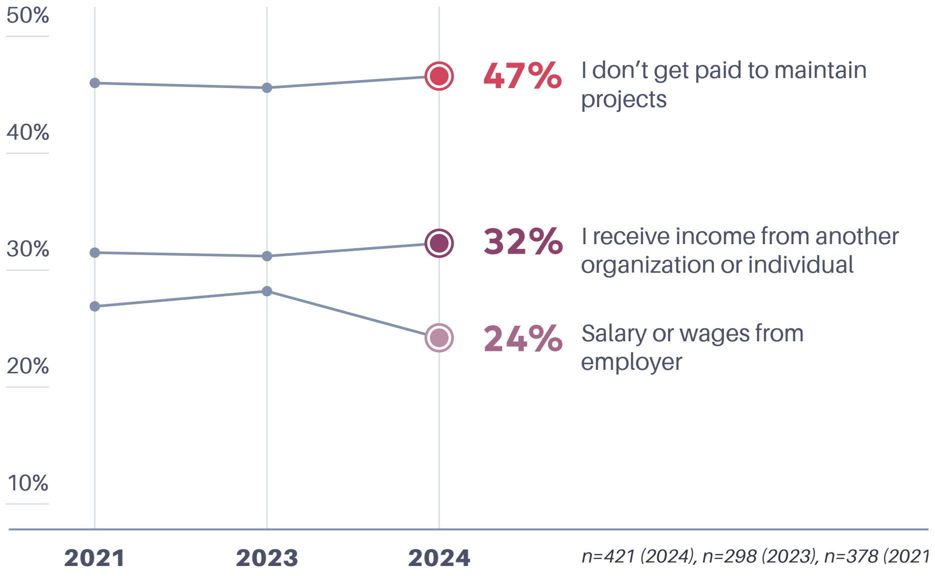
“I don’t get paid to maintain projects” vs. “I’m an unpaid hobbyist”

Before we delve too deeply into the answers, here’s an aside for any survey nerds like us (we know you are out there!). The way questions are worded or otherwise set up can have a large impact on the answers, and we have a good illustration of that here.

In the earlier question, we forced a single response, versus in this question, we let maintainers choose multiple answers that applied to them. Interestingly, even though the earlier question taught us that 60% of maintainers consider themselves unpaid hobbyists, when prompted with a set of potential income sources, only 47% report that they do not get paid to maintain projects. We asked both of these questions in our previous survey as well (and this particular question in all three surveys), and the results have stayed remarkably consistent over time, as the chart on the next page shows.

Who’s paying the maintainers?

Which of the following describe the source of your maintainer income? (Choose all that apply)



n=421 (2024), n=298 (2023), n=378 (2021

So what gives? Why is there a 13-percentage-point (60% vs. 47%) swing on what seems like a very similar datapoint? Because of the way this question is worded, it likely comes down to the *amount* of money maintainers are being paid. Some maintainers who consider themselves unpaid hobbyists probably only receive a nominal amount of income and not enough for them to begin to think of themselves as semi-professional or professional maintainers. Even if they technically received some payment for their work, it isn’t enough to make them identify as a professional or semi-professional maintainer. Or perhaps being prompted with some potential income sources or being able to choose multiple responses made them provide answers with a more nuanced perspective.

Who *is* paying the maintainers?

The percentage of maintainers who report they don’t get paid to maintain projects (47%) has stayed consistent in all three surveys (46% in both previous surveys), as has the percentage who report receiving income from another organization or individual (e.g. Tidelift, GitHub Sponsors, foundations, etc.) to maintain a project (32% in this survey, 31% in the previous survey, and 32% in the survey before that).

But we *did* see a slight drop in the percentage of maintainers who consider maintenance to be an explicit part of their paid job responsibilities. In this year’s survey, only 24% cited salary or wages from their employer as a source of maintainer income, which is less than the 2023 survey (28%) and the 2021 survey (27%), but still close enough that the difference is likely not statistically significant.

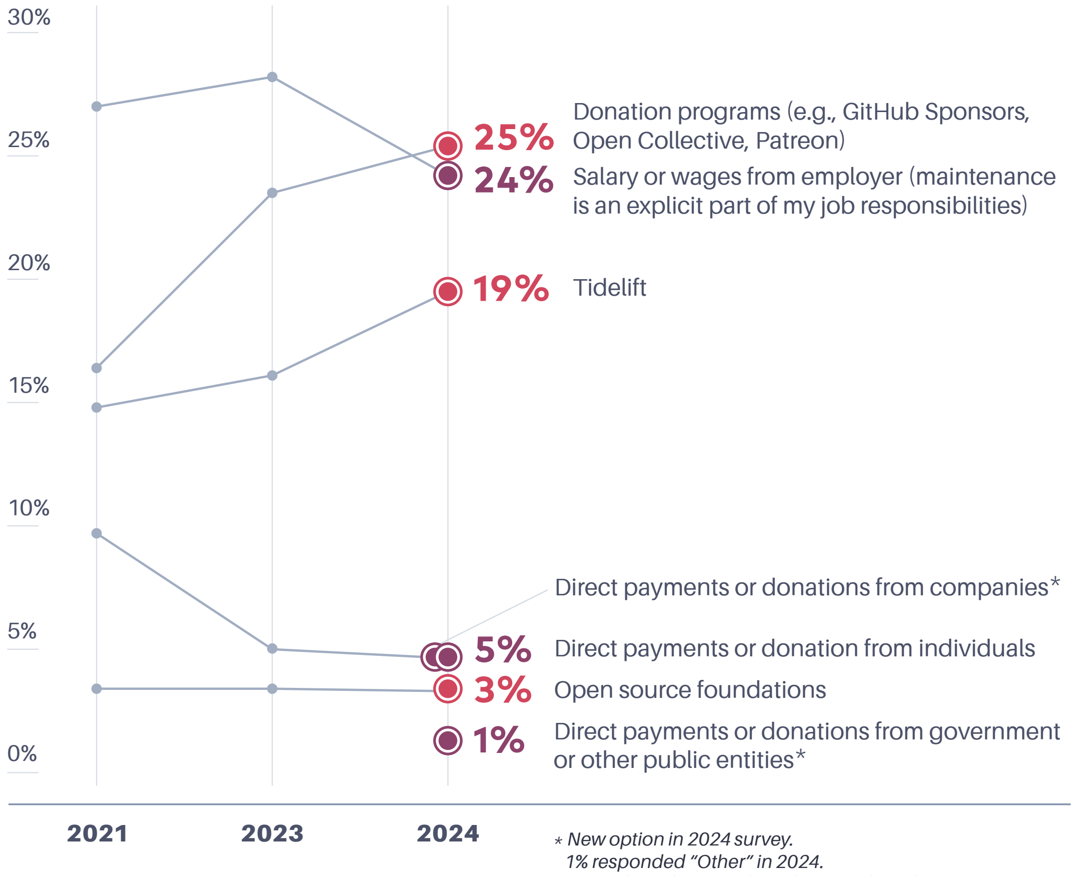
Things get more interesting when we look at the follow-up answers from maintainers who report receiving income from another organization or individual to maintain a project. As we did in our previous surveys, we asked maintainers where that income was coming from. The percentage of maintainers receiving income from donation programs like GitHub Sponsors has risen slightly, from 16% in 2021, to 24% in 2023, and to 25% in 2024. As we noted earlier, 24% of maintainers receive income from their employer because maintenance is explicitly part of their job responsibilities (and, again, that is down from 28% and 27% in previous surveys).

At 19%, Tidelift is the third highest reported source of income, up from 2023 (16%) and 2021 (15%).

Side note: in case you were wondering how Tidelift's maintainer partners figured into the results here, the large majority (70%) of survey respondents are not Tidelift maintainer partners, which is consistent with the 2023 survey (70%) and 2021 survey (68%).

Who's paying the maintainers? Donation programs, employers, and Tidelift

Which of the following describe the source of your maintainer income? (Choose all that apply) Maintainers with third-party income were also asked, "From which of the following sources do you receive non-employer-related income for open source maintainer work?"



** New option in 2024 survey.*
1% responded "Other" in 2024.
n=421 (2024), n=298 (2023), n=378 (2021)

The number of maintainers who report getting paid directly by companies (not their employer), governments, individuals, or foundations, is much lower than any of the top three income sources.

In this survey only 5% of maintainers report receiving income directly from companies (this answer choice was not an option in previous years). Another 5% report getting direct payments from individuals, which is steady compared to 2023, but much lower than the 10% of maintainers receiving this type of income in 2021. And only 3% of maintainers report that they have received income from open source foundations, which has remained steady across all three surveys (it may be surprising to some that this percentage is not higher).

Because governments around the world have taken a greater interest in open source software security over the past few years in the wake of prominent security incidents like SolarWinds, Log4Shell, and xz utils, we asked maintainers in our latest survey whether they were receiving income directly from governments or other public entities. But to date, this income source is a non-factor, with only 1% of maintainers reporting receiving direct payments from governments or other public entities.

As you review these findings, don't lose sight of the fact that **none of these sources of maintainer income are being reported by more than one quarter of maintainers**. This is one of several warning signs you'll find throughout this year's report that show we have a lot of work to do to ensure the amazing open source maintainers we all depend on have the financial support they need to keep their projects healthy, safe, and secure. ■

HEADLINE #4

Maintainers are spending 3× more time on security than they did a few years ago

We are always interested to learn more about how maintainers spend their time. So for this year's survey, we asked them, as we had in our 2021 survey (we skipped the question in the 2023 survey), to break down the amount of time per month they spend in the following areas on their projects:

- Security work (including fixing vulnerabilities and issuing patches, code scanning, dealing with insecure dependencies, complying with security best practices, and responding to new security research reports)
- Day to day maintenance work (including writing documentation, reviewing PRs from contributors, general dependency management, reviewing and responding to issues, and removing technical debt)
- Building new features (i.e. writing and testing new code)
- Seeking financial support and sponsors
- Other

The total amount of time had to equal 100%, and we included a slightly longer set of categories to choose from in our 2021 survey that we've combined into "other" for the sake of simplicity here (those additional categories included "Marketing and external communication," "Meetings, management, and operations of the project," and "Guiding the project's strategic direction," none of which accounted for more than 5% of their time).

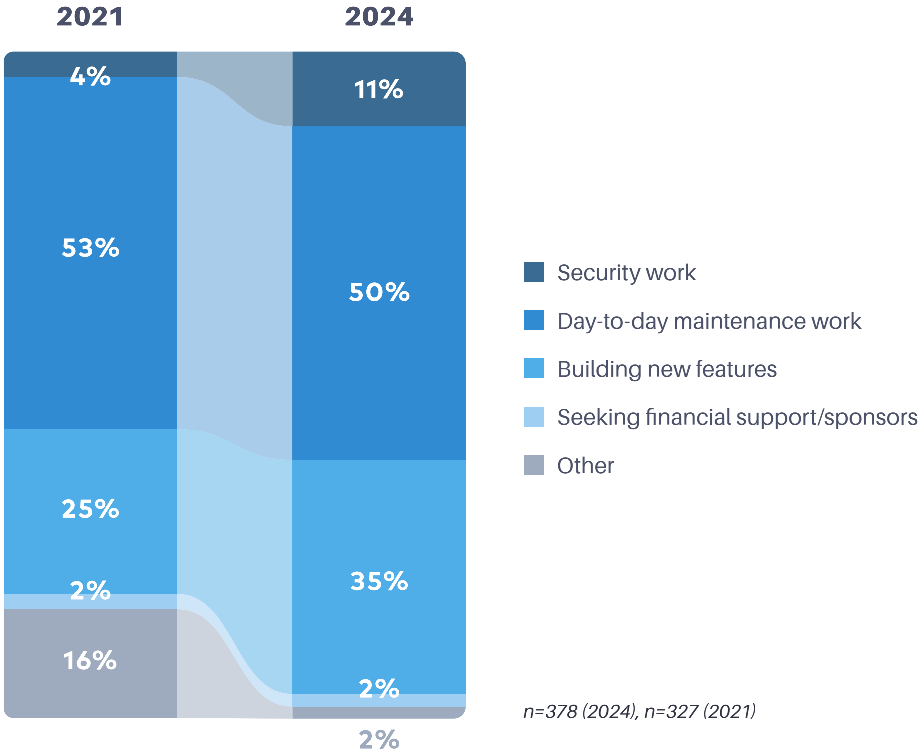
How do maintainers spend their time in 2024 vs. 2021?

While the percentage of time spent on day-to-day maintenance work stayed pretty consistent between the 2021 and 2024 surveys, from 53% in the previous survey to 50% today, the other percentages changed quite a bit (see chart on the following page).

Perhaps the most significant change was that maintainers now report they are spending almost 3× more time (11%) on security work than they reported in 2021 (4%). While this could be in part a factor of the changes to the category choices we provided this year, it is also not surprising, given that maintainers are also seeing increasing demands for their time from corporate users of their projects, security companies giving them potential vulnerabilities to investigate, and pressure to comply with new security requirements and initiatives like the OpenSSF Scorecard and the NIST Secure Software Development Framework, among others.

How maintainers spend their time: 2024 vs. 2021

Break down the amount of time spent per month on each of the following areas for the projects you maintain (Total must add up to 100%)



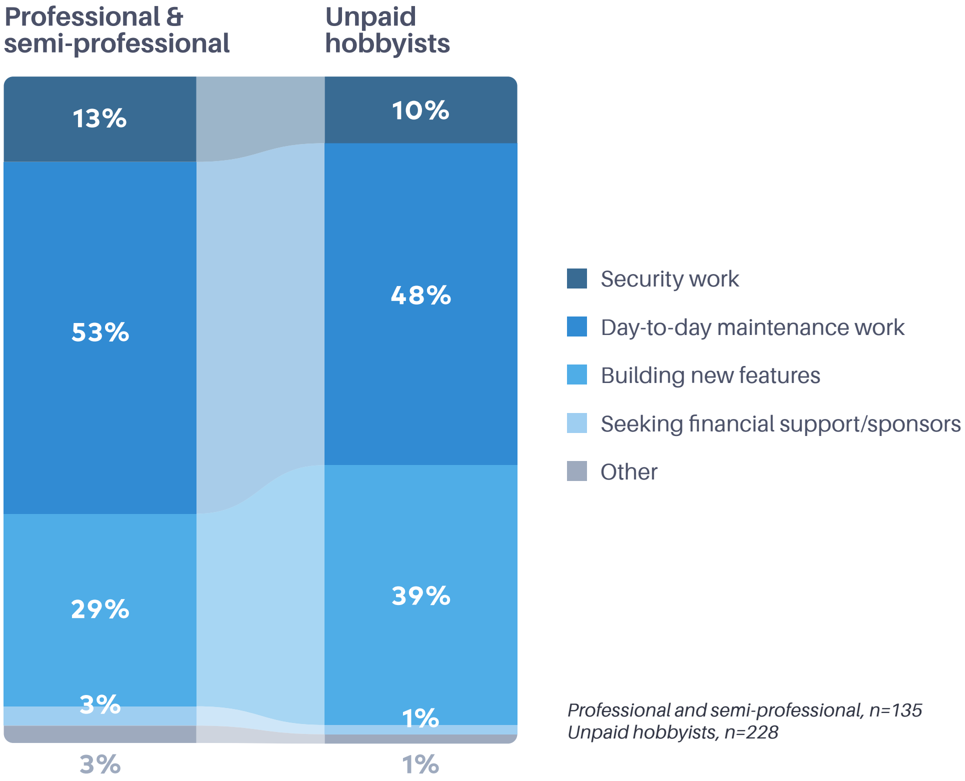
n=378 (2024), n=327 (2021)

Building new features also increased significantly as a percentage of the time maintainers spend on project maintenance work, going from 25% to 35% between 2021 to 2024.

We also took a look at the same data to see if maintainers who consider themselves to be professional or semi-professional maintainers spend their time differently than those who consider themselves unpaid hobbyists.

Paid maintainers spend more time on security and maintenance, less on new features

Break down the amount of time spent per month on each of the following areas for the projects you maintain (Total must add up to 100%):



Professional and semi-professional, n=135
Unpaid hobbyists, n=228

Professional and semi-professional maintainers spend slightly more time on security (13% vs. 10%) and maintenance (53% vs. 48%) work than unpaid hobbyist maintainers, which comes at the expense of their having as much time to build new features (29% vs. 39%).

Regardless of whether they are paid or unpaid, all maintainers need to make tradeoffs with the limited time they have to work on their projects. For maintainers of larger, more established projects with many users, we would not be surprised if the percentage of time they need to spend on security and maintenance work continues to increase over time as the number of requirements and complexities they are expected to manage rises with it.

HEADLINE #5

Maintainers are much more likely to align with the OpenSSF Scorecard when they are paid for their work

As we shared in our previous finding, maintainers are being asked to do more work to comply with increasingly complex security requirements from industry and government. We started following this trend in our previous maintainer survey, and were able to collect some additional new data points to share in this year's survey report as well.

We repeated a question we'd included in our previous survey, where we asked maintainers to report whether they were aware of some of the most common industry security standards or initiatives. In our previous survey, we'd asked about the NIST Secure Software Development Framework (SSDF), the OpenSSF Scorecard, and the Supply Chain Levels for Software Artifacts (SLSA) Framework, and this year we added the Secure by Design pledge that was initiated by CISA (the Cybersecurity and Infrastructure Security Agency) of the U.S. government.

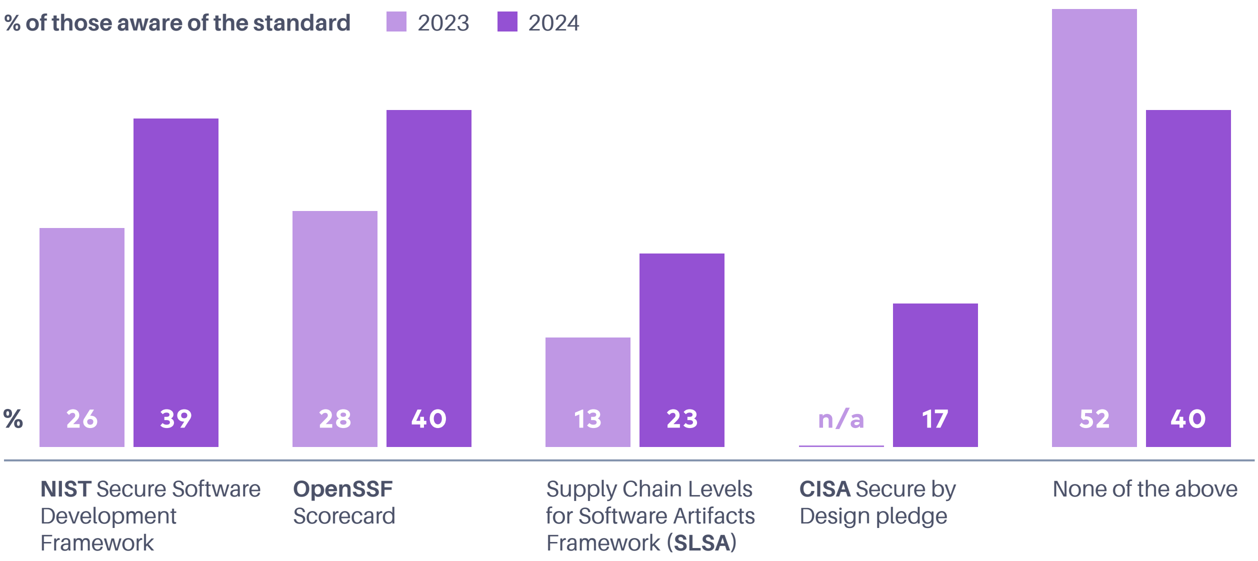
How aware are maintainers of the most common industry security standards and initiatives?

Across the board, the percentage of maintainers who are aware of these industry standards and initiatives has grown. The initiative with the highest awareness among maintainers is the OpenSSF Scorecard project, with 40% of maintainers being aware of it, up from 28% in our previous survey. This is followed closely by the NIST SSDF, with 39% awareness, up from 26% in our previous survey.

More maintainers are also aware of the SLSA framework (23%) this year, compared to only 13% when we asked about it in 2023. And in our first year including it, 17% of maintainers were aware of the CISA Secure by Design pledge. The percentage of maintainers that were not aware of any of these initiatives decreased from 52% in 2023 to 40% this year, as these initiatives continued to gain adoption and traction (see chart on the following page).

More maintainers are aware of common industry security standards in 2024 than 2023

Which of the following industry standards or initiatives are you aware of? (Choose all that apply)



2% reported "Other" for both 2023 and 2024. n=378 (2024); n=292 (2023)

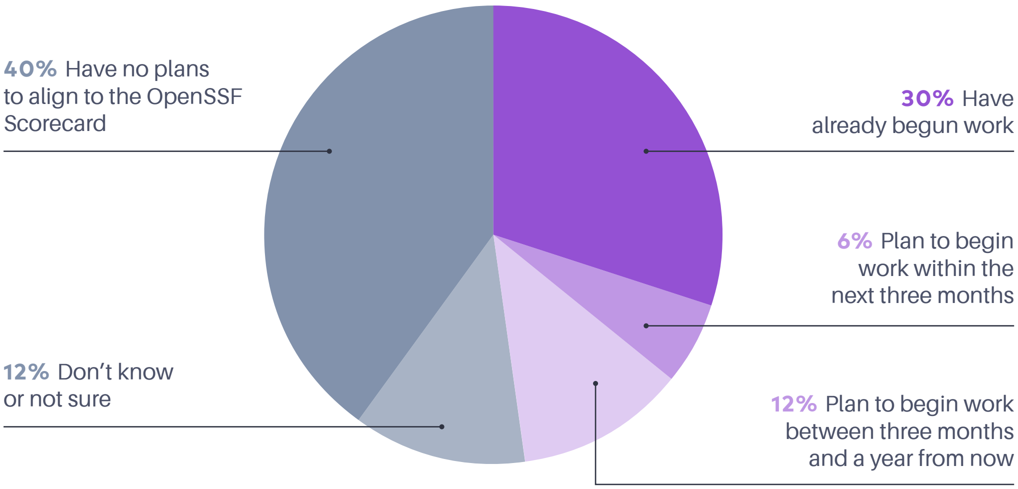
Are maintainers aligning their projects with OpenSSF Scorecard requirements?

We were particularly interested in the responses regarding the OpenSSF Scorecard project, which is becoming a security standard benchmark for many enterprise organizations. So we asked maintainers who had indicated that they were aware of the OpenSSF Scorecard project if they already have begun or plan to begin work to ensure their projects align with its requirements.

Thirty percent of these maintainers have already begun work to ensure their projects align with the requirements of the OpenSSF Scorecard, while another 6% plan to begin work in the next three months, and 12% plan to begin work between three months and one year from now. A full 40% of maintainers currently have no plans to align to the OpenSSF Scorecard.

30% of maintainers aware of the OpenSSF Scorecard have already begun the work to align to it

Respondents aware of the OpenSSF Scorecard were asked, "Have you already or do you plan on beginning the work needed to ensure your projects align with the requirements of the OpenSSF Scorecard?"



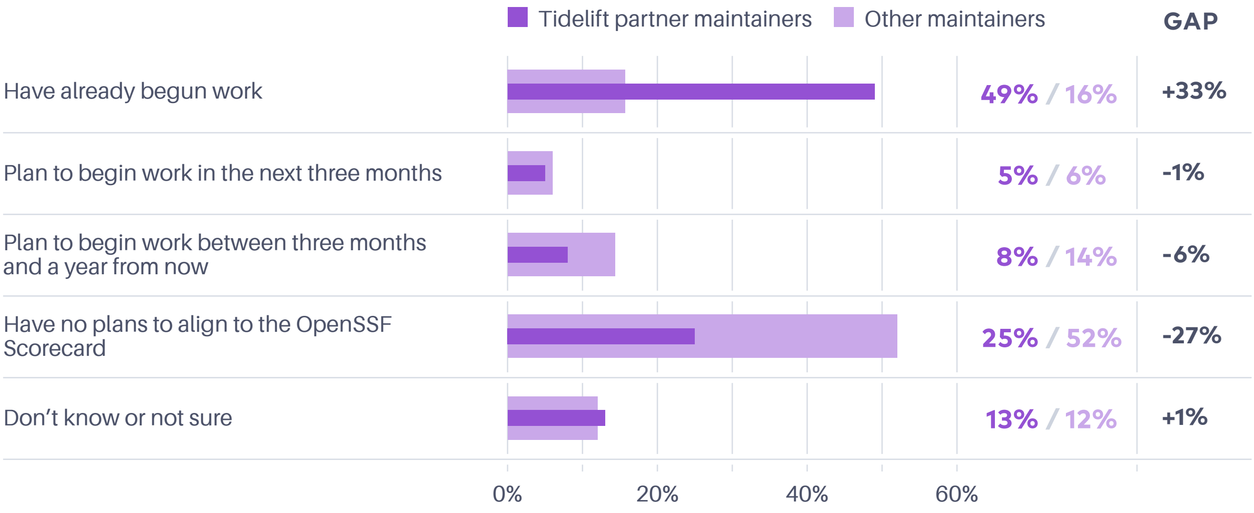
n=146

The data gets really interesting when you compare the maintainers who have partnered with Tidelift to those who have not. (Tidelift partners with open source maintainers and pays them to implement industry-leading secure software development practices—like many of those found in the OpenSSF Scorecard—validate the practices they follow, and then contractually commit to continue these practices into the future.)

Nearly half of Tidelift-partnered maintainers aware of the OpenSSF Scorecard (49%) have already begun the work to ensure their projects align with its requirements, which is three times the percentage of maintainers not partnered with Tidelift who have done so (16%). **This is about as clear evidence as any we have seen that paying maintainers delivers results when it comes to implementing better secure software development practices**. Conversely, over half (52%) of maintainers who are not partnered with Tidelift have no plans to align to the OpenSSF Scorecard.

Tidelift-partnered maintainers are much more likely to have begun the work to align with OpenSSF Scorecard standards

Respondents aware of the OpenSSF Scorecard were asked, “Have you already or do you plan on beginning the work needed to ensure your projects align with the requirements of the OpenSSF Scorecard?”



Tidelift maintainer partners, n=63; Other maintainers, n=83

There are several reasons why maintainers might not want to align their projects with the OpenSSF Scorecard. The simplest reason is that it is a lot of work, and they are not being paid to do it. We suspect this may be why many maintainers are not aligning with the OpenSSF Scorecard yet.

Another reason might be that they just aren’t familiar enough with all of the requirements to make the commitment. Or some maintainers may not agree with all of the OpenSSF Scorecard requirements or the requirements may not all apply to their projects and/or ecosystem. For example, the scorecard includes a binary artifacts check, but virtually nothing in the npm ecosystem distributes binaries, so JavaScript maintainers may not be aligning their projects to scorecard checks like these they do not feel are relevant.

Notwithstanding cases like these, the stark differences between how Tidelift-partnered maintainers answered this question about the OpenSSF Scorecard and how non-partnered maintainers answered the same question are **a strong general signal that paying open source maintainers to implement secure software development practices can be an effective way for organizations to improve the security of the open source software supply chain they rely on**. In our next finding, we’ll get into even more detail about the security practices maintainers are willing to implement when they are paid for their work. ■

HEADLINE #6

Paid maintainers are significantly more likely to implement critical security practices than unpaid maintainers

In our previous finding, we extensively covered how many maintainers are aware of common open source security initiatives like the OpenSSF Scorecard project and the NIST Secure Software Development framework. Part of the usefulness of initiatives like these is that they outline lists of secure software development practices that maintainers can follow to keep their projects safe and secure.

For this part of the survey, we wanted to look more closely at specific security practices like the ones found in the NIST SSDF, OpenSSF Scorecard, or required for Tidelift-partnered maintainers so we could learn which of them maintainers already have in place and which they would consider implementing in the future.

Security practices most implemented by maintainers today

First, we asked a question similar to one we had asked previously in the 2023 survey:

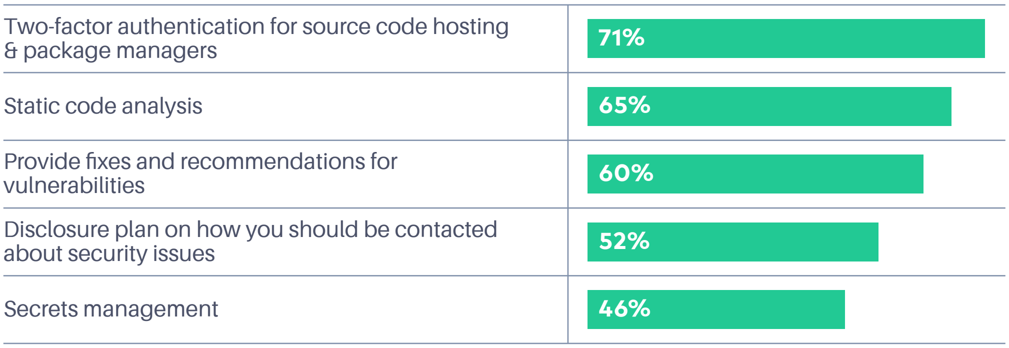
> Which of the following security practices have been implemented for most or all of the projects you maintain?

Except this time, we provided a much longer set of options than we'd included in our last survey. Of this new, longer list of common security practices, the one implemented by the highest percentage of maintainers was two-factor authentication for source code hosting and package managers (71%). Second was static code analysis (65%), and third was that they provide fixes and recommendations for vulnerabilities (60%). These were followed by a security disclosure plan on how they should be contacted about security issues (52%) and secrets management (46%) (see chart on the following page).

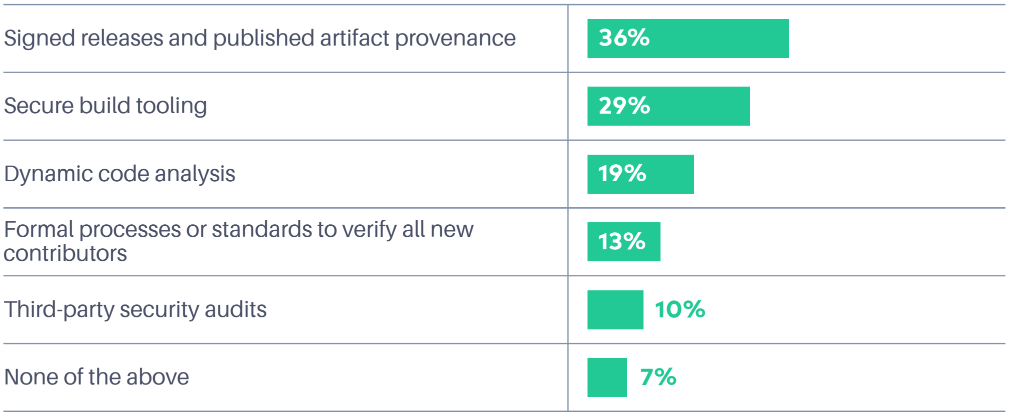
Which security practices are implemented most often by maintainers today?

Which of the following security practices have been implemented for most or all of the projects you maintain? (Choose all that apply)

TOP 5



OTHER



n=368

Maintainer security practices: 2024 vs. 2023

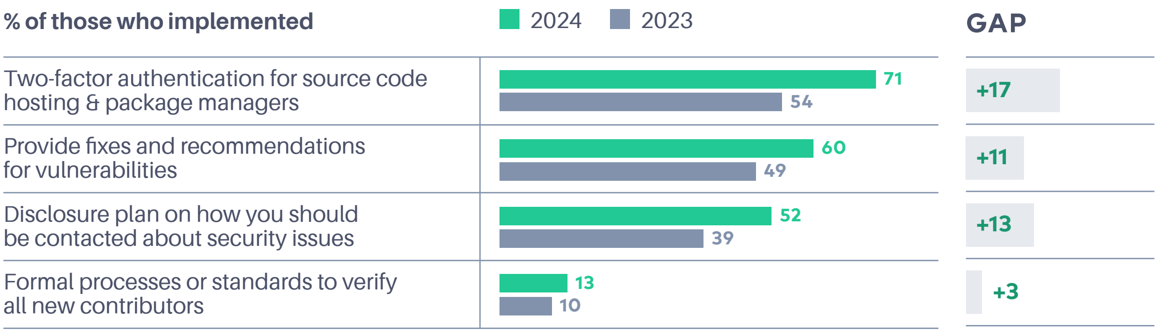
For those practices specifically asked about in both 2023 and 2024, we wanted to compare against our previous results to see if more maintainers are implementing the practices today than in 2023, and in fact they are.

Two-factor authentication was only implemented by 54% of maintainers in 2023, but now is being implemented by 71% of maintainers (+17%), which makes sense now that GitHub has made two-factor authentication basically mandatory for projects hosted on its platform. This is an interesting signal that for some classes of security practices, centralized infrastructure changes might be part of the solution.

Only 49% of maintainers were providing fixes and recommendations for vulnerabilities in 2023, and that percentage has risen to 60% today (+11%). And 39% were implementing a security issue disclosure plan in 2023 and the percentage is 52% today (+13%).

Maintainers are implementing common security practices more often than they were in 2023

Which of the following security practices have been implemented for most or all of the projects you maintain? (Choose all that apply)



n=368 (2024); n=280 (2023)

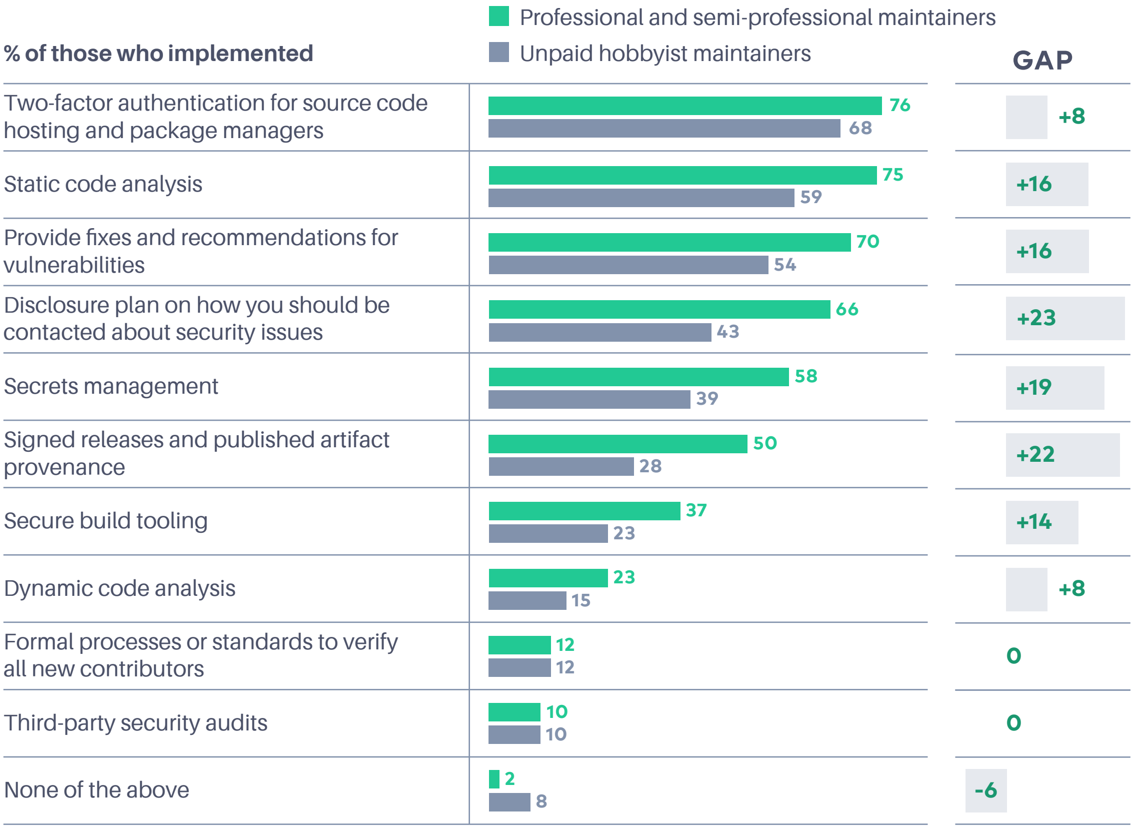
Do paid maintainers implement more security practices than unpaid maintainers?

Next, we wanted to see if maintainers who are being paid for their work are more likely to implement critical security practices than those who are not paid, and nearly across the board they are. In fact **when you look across all of the security and maintenance practices we asked about, paid maintainers are 8-26 percentage points (or, on average 55%) more likely to implement the practices than unpaid maintainers**. We'll talk more about maintenance practices in the next finding, but for now, we'll start with security practices.

For the three most implemented security practices, two-factor authentication (+8%), static code analysis (+16%), and providing fixes and recommendations for vulnerabilities (+16%), paid maintainers are significantly more likely to have implemented the practices than unpaid maintainers. The gaps get even more pronounced among the next set of practices. Paid maintainers were much more likely to have implemented a security disclosure plan (+23%), implemented secrets management (+19%), and have signed releases and published artifact provenance (+22%).

Paid maintainers implement more security practices than unpaid maintainers

Which of the following security practices have been implemented for most or all of the projects you maintain? (Choose all that apply)



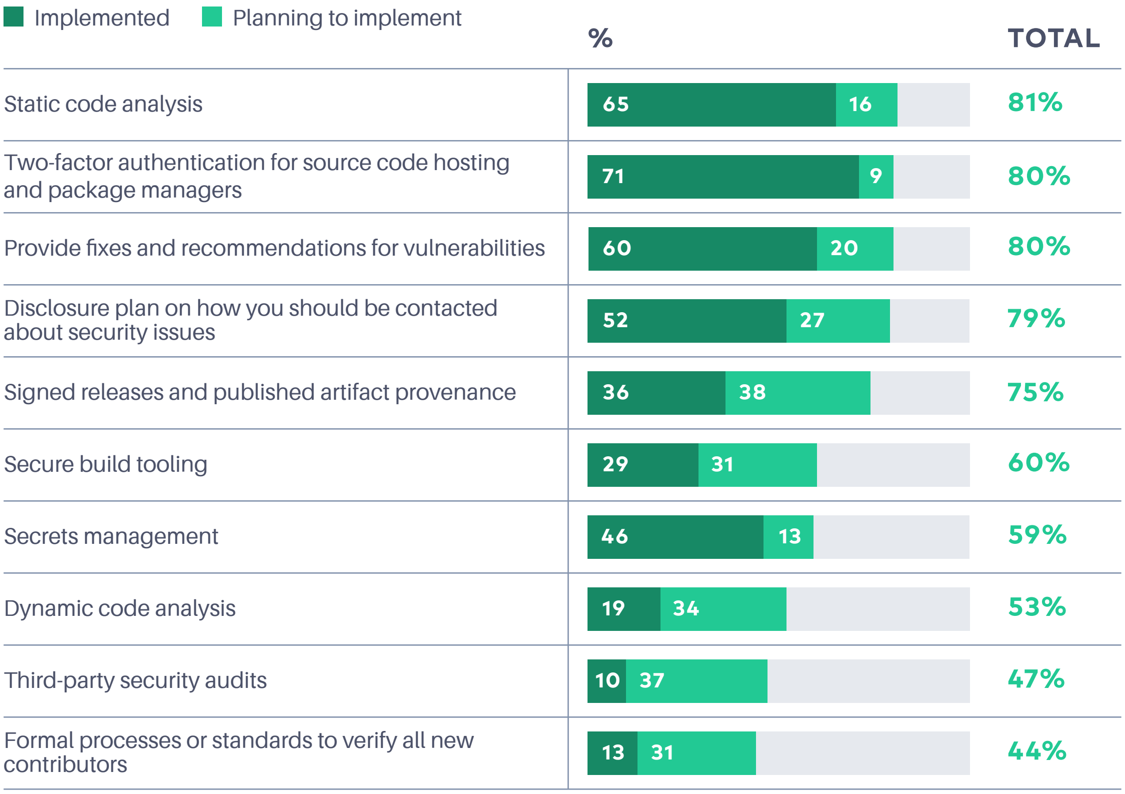
Professional or semi-professional maintainers, n=132; Unpaid hobbyist maintainers, n=221

What are the top security practices paid maintainers would implement?

Finally, we took the practices that individual maintainers reported they were not implementing today, and we asked them which of these practices they would consider implementing *if they were paid for the work*. As the chart below shows, when you combine the practices maintainers are *already completing* with the additional practices they would be *willing to complete* if they were being paid for the work, a roadmap for what security practices we might be able to positively impact by paying maintainers begins to emerge.

What security practices would maintainers implement if they were paid for the work?

Which of the following security practices have been implemented for most or all of the projects you maintain? (Choose all that apply) Of the security practices that have not been implemented for most or all of the projects that you maintain, which would you consider implementing in the future if paid to do so? (Choose all that apply)



n=368

For example, only a small percentage of maintainers implement dynamic code analysis (19%), formal processes or standards to verify new contributors (13%), and third-party security audits (10%). But if they were paid, the number of maintainers that would at least consider implementing these less common, but still critical, practices roughly *triples* to 53%, 44%, and 47% respectively.

Looking at the top five security practices in the chart on the previous page, it becomes clear that **over three quarters of all maintainers would be at least willing to implement the most common security practices** like static code analysis (81%), two-factor authentication (80%), providing fixes and recommendations for vulnerabilities (80%), providing a security disclosure plan (79%), and providing signed releases and published artifact provenance (75%) **if they were being paid for the work**.

This is an exciting finding, because if you look at it the other way, it shows that many maintainers are not being held back from completing many common security tasks due to a lack of understanding or willingness to implement these practices.

It is that implementing these practices, and keeping them in place over time, requires a lot of work. **Maintainers are clearly telling us that they are willing to do the work required to secure their projects—but they aren't willing to do it for free.** 🟩

HEADLINE #7

Paid maintainers do more maintenance and documentation work than unpaid maintainers

In our previous finding we shared results about the security practices maintainers have implemented for their projects or would be willing to implement if they were paid for the work. As we did last year, we also asked about a common set of maintenance and documentation practices to better understand which they have already implemented or would be willing to implement as well.

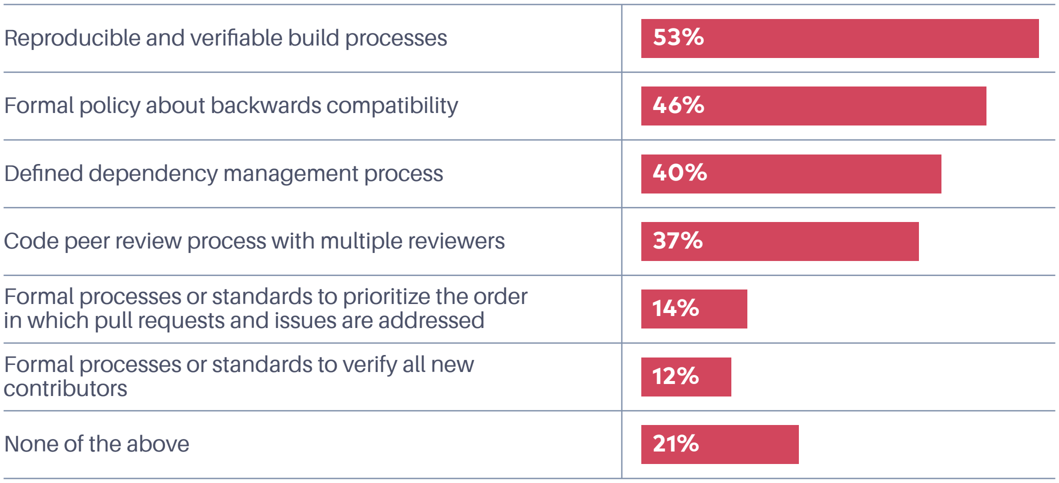
Maintenance practices most implemented by maintainers today

First, we asked maintainers about the maintenance practices they have implemented today. Only one of the maintenance practices we asked about has been implemented by more than half of maintainers: providing reproducible and verifiable build processes (53%).

The next most common practice was having a formal policy about backwards compatibility, which has been implemented by 46% of maintainers. This was followed by having a defined dependency management process (40%) and having a code peer review process with multiple reviewers (37%).

Which maintenance practices are implemented most often by maintainers today?

Which of the following maintenance practices have been implemented for most or all of the projects you maintain?



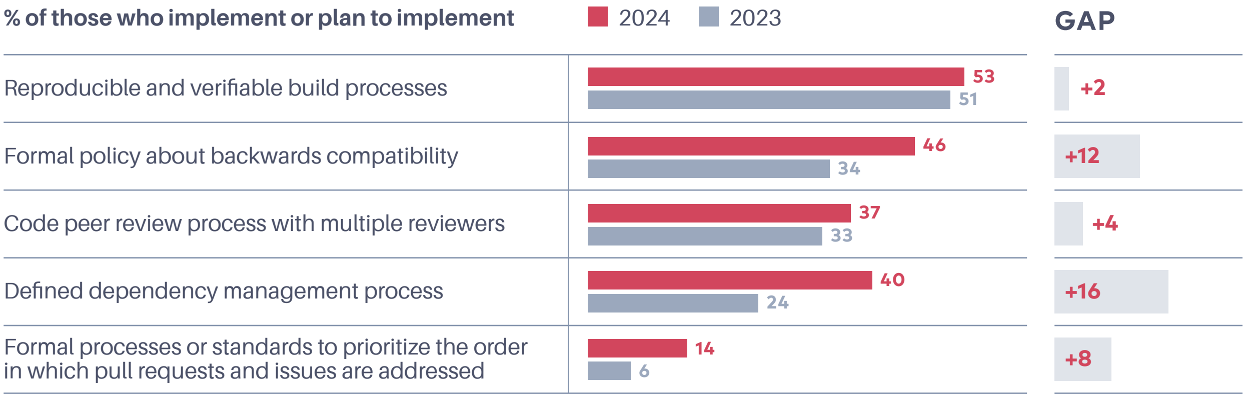
n=359

For the maintenance practices covered in our 2023 survey, we also wanted to see if maintainers are implementing more maintenance practices today. Just like with security practices, adoption has risen across the board.

The practices with the biggest increases were having a defined dependency management process (+16%, to 40% in this year's survey) and having a formal policy about backwards compatibility (+12%, to 46% in this year's survey).

Maintainers are implementing common maintenance practices more often than they were in 2023

Which of the following maintenance practices have been implemented for most or all of the projects you maintain? (Choose all that apply) Of the maintenance practices that have not been implemented for most or all of the projects that you maintain, which would you consider implementing in the future if paid to do so? (Choose all that apply)

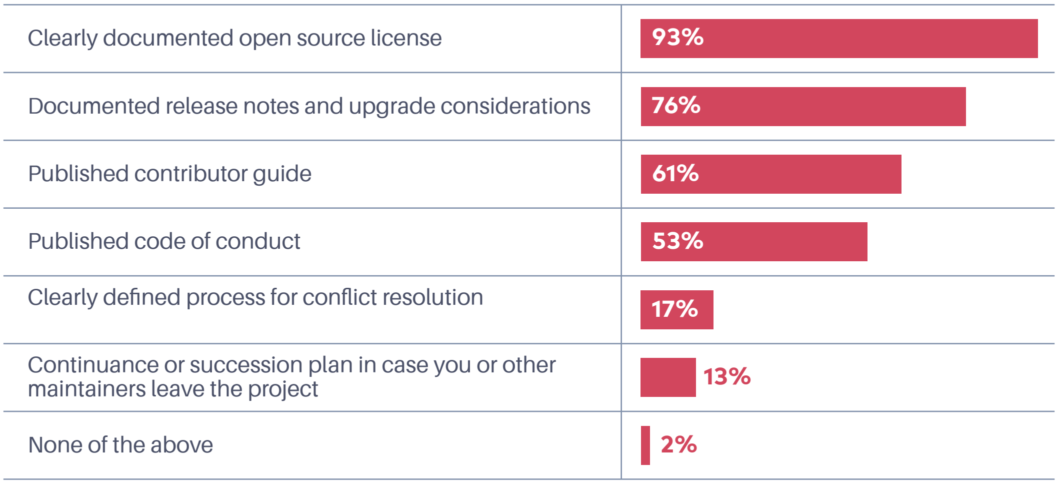


n=359 (2024); n=280 (2023)

We also asked maintainers about their common documentation practices. The top four practices we asked about are all implemented more often and by more maintainers than the top maintenance practice (as discussed above). First was having a clearly documented open source license, which almost all maintainers (93%) do today. Second was having documented release notes and upgrade considerations, which 76% of maintainers provide today. Third is publishing a contributor guide, which 61% of maintainers do today. And fourth is having a published code of conduct, which 53% of maintainers provide today (see chart on the following page).

Which documentation practices are implemented most often by maintainers today?

Which of the following documentation practices have been implemented for most or all of the projects you maintain?

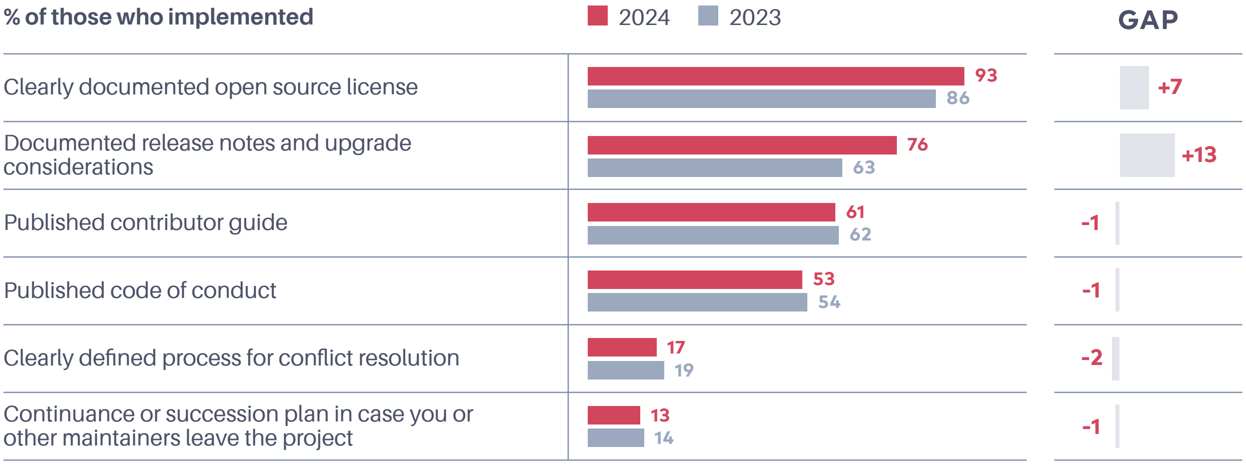


n=353

As with the maintenance practices, we also asked about many of these documentation practices in the 2023 survey. Several of these stayed more stable year over year, although the percentage of maintainers who have implemented the top two documentation practices, having a clearly documented open source license (+7% to 93% this year) and having documented release notes and upgrade considerations (+13% to 76% this year), both increased.

Only slight changes in documentation practices from 2023 to 2024

Which of the following documentation practices have been implemented for most or all of the projects you maintain?



n=353 (2024); N=280 (2023)

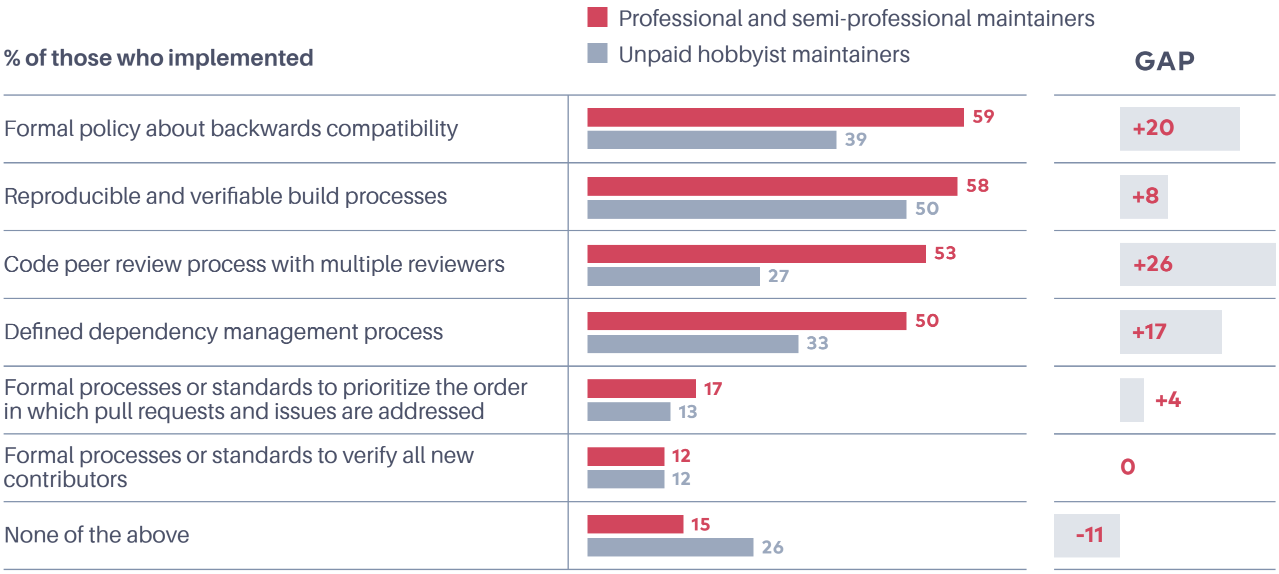
Do paid maintainers implement more maintenance and documentation practices than unpaid maintainers?

Next, we broke down the percentages of paid (professional and semi-professional) maintainers and unpaid hobbyist maintainers who complete these practices today. As was the case with common security practices, paid maintainers are much more likely to complete more common maintenance and documentation practices than unpaid maintainers.

On maintenance practices, the biggest gap between paid and unpaid maintainers was for having a code peer review process with multiple reviewers, which 53% of paid maintainers are implementing today (+26% above unpaid maintainers). Next was having a formal policy about backwards compatibility, which 59% of paid maintainers are implementing today (+20% above unpaid maintainers).

Paid maintainers implement more maintenance practices than unpaid maintainers

Which of the following maintenance practices have been implemented for most or all of the projects you maintain? (Choose all that apply)

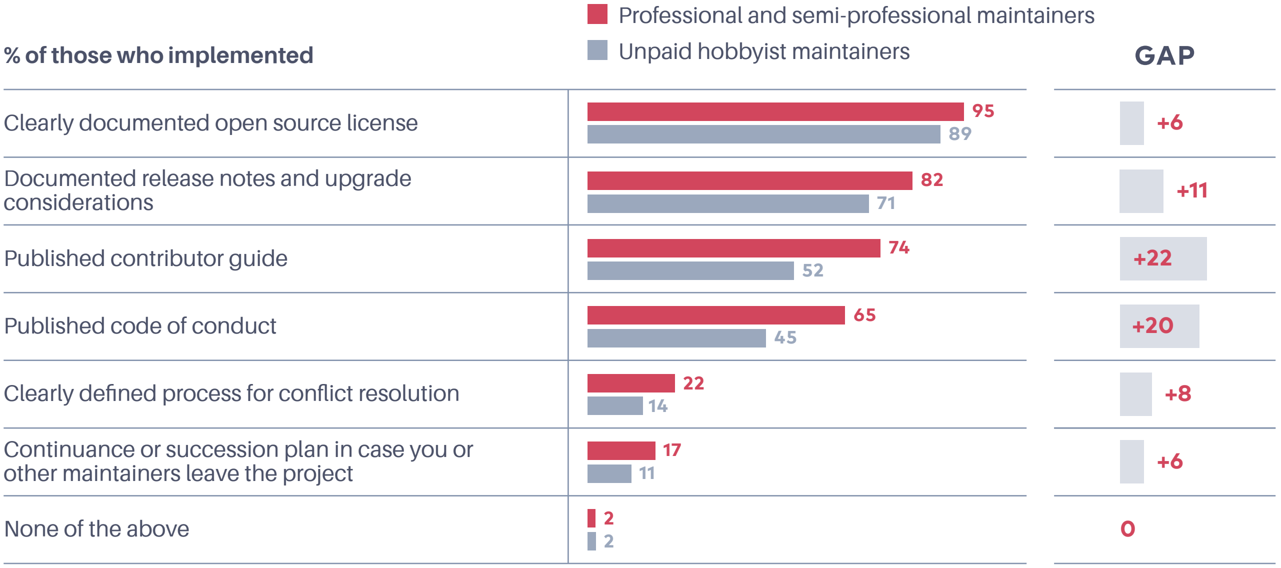


Professional or semi-professional maintainers, n=129; Unpaid hobbyist maintainers, n=216

Across the board, paid maintainers also currently implement more documentation practices. For the practices we asked about, the biggest gaps between paid and unpaid maintainers were for having a published contributor guide, which 74% of paid maintainers are providing (+22% above unpaid maintainers) and having a published code of conduct, which 65% of maintainers are providing (+20% above unpaid maintainers).

Paid maintainers implement more documentation practices than unpaid maintainers

Which of the following documentation practices have been implemented for most or all of the projects you maintain? (Choose all that apply)



Professional or semi-professional maintainers, n=128; Unpaid hobbyist maintainers, n=211

What maintenance and documentation practices would maintainers implement if they were paid for the work?

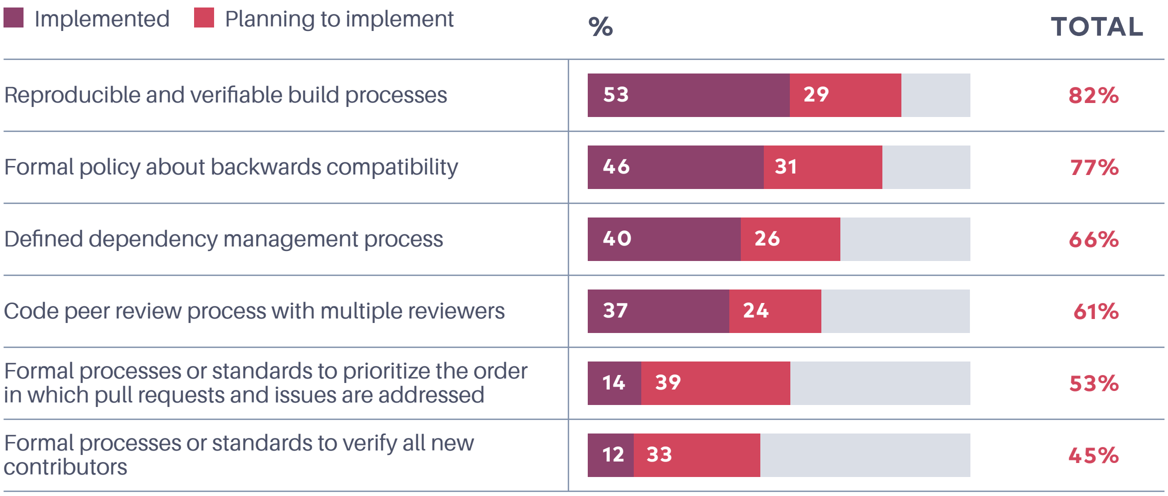
Finally, as we did with security practices in our previous finding, we combined the percentage of maintainers who reported that they had already implemented common maintenance and documentation practices with the percentage of maintainers who indicated that they would be willing to implement these practices if they were paid for their work. This gives us a roadmap for what maintenance and documentation practices we might be able to positively impact by paying maintainers.

In the case of maintenance practices, we could expect that most maintainers would provide reproducible and verifiable build processes (82%) and formal policies around backward compatibility (77%) if they were paid, and about two-thirds of maintainers would also provide a defined dependency management process (66%) and a code peer review process with multiple reviewers (61%) if they were paid.

Even more interestingly, the percentages for some less implemented practices virtually triple when you add in the maintainers who would implement them if they were paid. For example, having a formal process or set of standards to prioritize the order in which pull requests and issues are addressed would jump from 14% of maintainers who implement today to 53% if you include the maintainers who report they would complete the task if they were paid for it. And having a formal process or set of standards to verify all new contributors would jump from 12% to 45%.

What maintenance practices would maintainers implement if they were paid for the work?

Which of the following maintenance practices have been implemented for most or all of the projects you maintain? (Choose all that apply) Of the maintenance practices that have not been implemented for most or all of the projects that you maintain, which would you consider implementing in the future if paid to do so? (Choose all that apply)



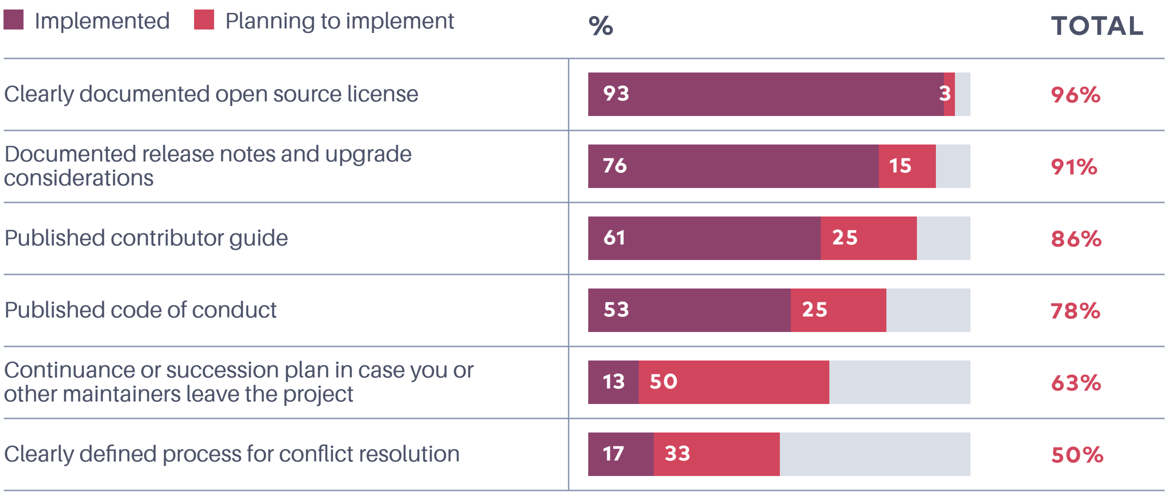
n=359

For common documentation practices, virtually all maintainers would provide a clearly documented open source license (96%) and documented release notes and upgrade considerations (91%) if they were paid (although these percentages were already high to begin with). Perhaps more interesting is the percentage of maintainers who would be willing to publish a contributor guide (86%) or a code of conduct (78%).

And as with the maintenance practices, two of the less-implemented documentation practices would become exponentially more common if the maintainers were paid. Only 13% of maintainers currently have a succession plan, but that percentage jumps to 63% when you include maintainers who would provide it if they were paid. Similarly, having a clearly defined process for conflict resolution is only implemented by 17% of maintainers today, but that percentage jumps to 50% when including those who would be willing to do the work to create a process if they were paid.

What documentation practices would maintainers implement if they were paid for the work?

Which of the following documentation practices have been implemented for most or all of the projects you maintain? (Choose all that apply) Of the documentation practices that have not been implemented for most or all of the projects that you maintain, which would you consider implementing in the future if paid to do so? (Choose all that apply)



n=353

Looking at all of the questions about security, maintenance, and documentation practices together, the findings are remarkably consistent, and perhaps unsurprising.

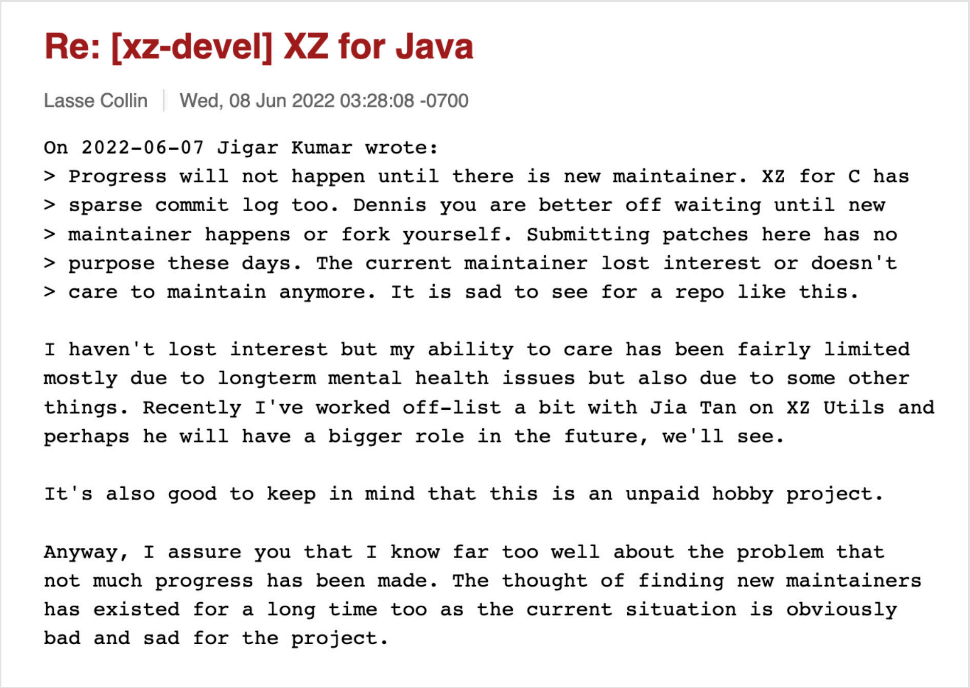
Paid maintainers already complete a lot more security, maintenance, and documentation work than unpaid maintainers. And there is **willingness on the part of maintainers to do even more**, but they have also made it abundantly clear: **if we want this important work done, we need to pay them for it.** 🟥

HEADLINE #8

Almost half of maintainers feel underappreciated and like the work is thankless

In our previous maintainer surveys, we've asked maintainers to tell us more about what they like and dislike about being an open source maintainer. We've gotten a good sense for what maintainers like about their work, so we decided not to ask about that again this year (feel free to review our previous reports if that subject interests you!).

But pressure on open source maintainers to do more continues to rise each year. And the multi-year hacking effort waged against xz utils maintainer Lasse Collin (sadly captured in this email exchange below) that was uncovered this year is a stark and timely reminder of the high costs maintainers sometimes pay to continue their work (we'll talk more about the xz utils hack in our next finding).



Re: [xz-devel] XZ for Java

Lasse Collin | Wed, 08 Jun 2022 03:28:08 -0700

```
On 2022-06-07 Jigar Kumar wrote:
> Progress will not happen until there is new maintainer. XZ for C has
> sparse commit log too. Dennis you are better off waiting until new
> maintainer happens or fork yourself. Submitting patches here has no
> purpose these days. The current maintainer lost interest or doesn't
> care to maintain anymore. It is sad to see for a repo like this.

I haven't lost interest but my ability to care has been fairly limited
mostly due to longterm mental health issues but also due to some other
things. Recently I've worked off-list a bit with Jia Tan on XZ Utils and
perhaps he will have a bigger role in the future, we'll see.

It's also good to keep in mind that this is an unpaid hobby project.

Anyway, I assure you that I know far too well about the problem that
not much progress has been made. The thought of finding new maintainers
has existed for a long time too as the current situation is obviously
bad and sad for the project.
```

So we decided to again ask the question about what maintainers dislike about their work to see if the answers had changed.

What do maintainers dislike most about their work?

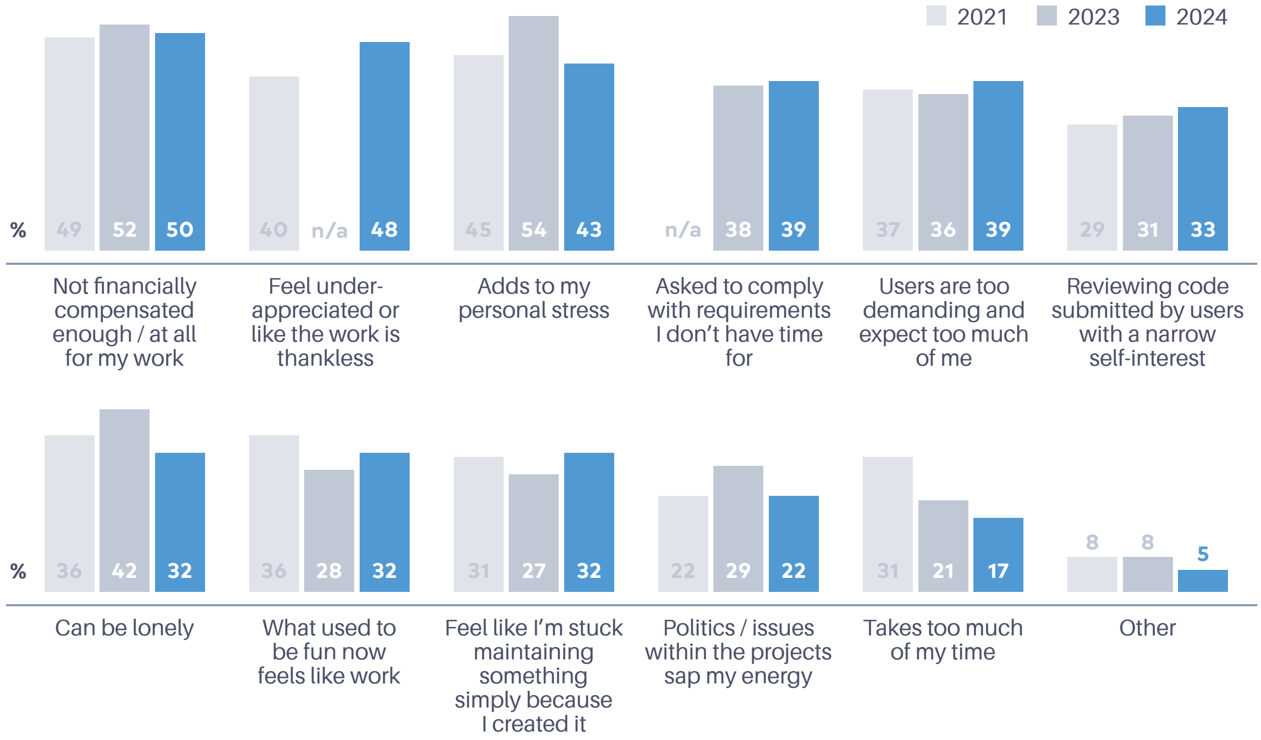
While the choices stayed mostly the same as in the last survey, we brought back one choice “feel underappreciated or like the work is thankless” that we did not ask about in 2023. And we’re glad we did. Since we last provided that as an option in 2021, when 40% of maintainers selected it, almost half (48%) of maintainers now feel that way about their work.

The only choice that, unsurprisingly, was selected by slightly more maintainers (50%) as something they dislike was “not financially compensated enough/at all for my work,” which has remained consistent in all three surveys (49% in 2021 and 52% in 2023).

Last year’s number one response “adds to my personal stress” dropped significantly from 54% to 43%, which we believe is because we did not include the option for “feel underappreciated or like the work is thankless” in 2023. This response actually tracks with where it was previously in the 2021 survey when we provided both options (43% this year vs. 45% in 2021). Still, even though it rated lower this year, “adding to my personal stress” placed a solid third as a thing maintainers dislike about their work.

Not being financially compensated enough and feeling underappreciated are the top things maintainers dislike about their work

What do you dislike about being an open source maintainer? (Choose all that resonate with you)



n=338 (2024); n=253 (2023); n=253 (2021)

Other selections that dropped significantly this year include "can be lonely," which dropped from 42% in 2023 to 32% this year, and "takes too much of my time" which dropped from 31% in 2021 to 21% in 2023 and 17% this year.

What maintainers dislike about their work in their own words

We also asked maintainers to tell us in their own words why they made the selections they did, and parsed the answers to reveal key themes. Many of the responses were related to time and work-life balance, while other top themes were lack of appreciation and support, user entitlement and demands, and financial challenges.

On the subject of time and work-life balance, one maintainer shared:

> "You end up doing a lot of stuff simply because it needed to be done, and nobody else was doing it. You feel you're responsible for it, the work just keeps piling up, and in the end, you're so busy you don't even have time to try to find co-maintainers."

Another maintainer complained about the sense of entitlement many users have.

> "Most users, even ones who require fixes, are not willing to roll up their sleeves to help. They just expect someone else to fix it for free."

Or, as another maintainer put it more bluntly:

> "The entitlement of the open source community is off the charts."

These demands can often seem callous and uncaring, as one maintainer described:

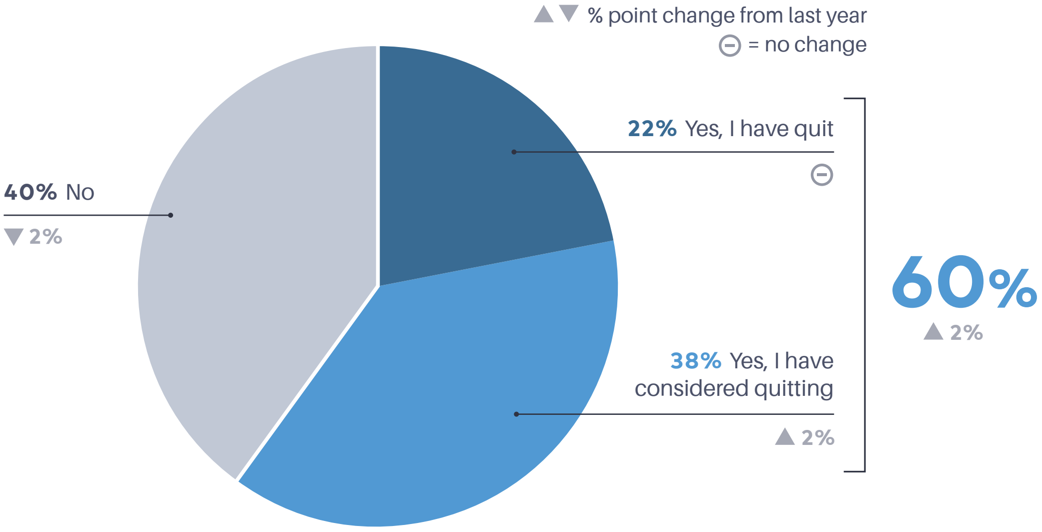
> "Users can be so entitled. 'Why haven't you merged/fixed this? This project is dead.' No, I have debts, a full-time job, a young family, my parent just died, and my wife has a serious medical issue. I have already sunk thousands of hours into this project, I don't have time to deal with this right now."

How many maintainers are quitting?

It's no wonder that year after year, our survey shows that more than half of maintainers have either quit or considered quitting their maintenance work. This year the percentage of maintainers who have either quit or considered quitting their work was 60%, which is consistent with the 58% in 2023 and 59% in 2021.

More than half of maintainers have quit or considered quitting their maintenance work

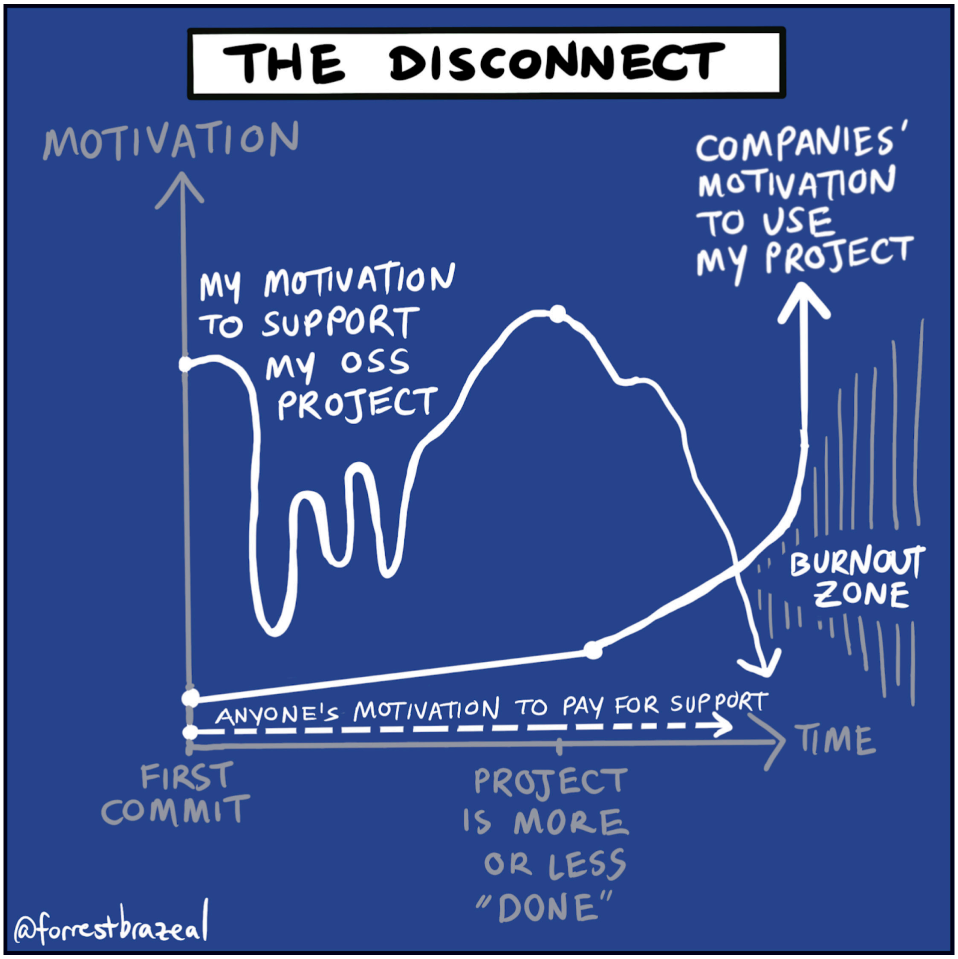
Have you quit or considered quitting maintaining a project?



n=350 (2024); n=265 (2023)

One maintainer summed up their feelings very succinctly:

> "Open source has powered a massive trillion-dollar injection of value into the world, the financial value of which has been reaped by large corporations, which on the whole give very little back to the ecosystem, not even appreciation, respect, or gratitude."


THE DISCONNECT
MOTIVATION
COMPANIES' MOTIVATION TO USE MY PROJECT
MY MOTIVATION TO SUPPORT MY OSS PROJECT
BURNOUT ZONE
ANYONE'S MOTIVATION TO PAY FOR SUPPORT
TIME
FIRST COMMIT
PROJECT IS MORE OR LESS "DONE"
@forrestbrazeal

As this Forrest Brazeal cartoon shows, **the disconnect between companies' expectations of open source and maintainer motivation to continue working on open source is extremely dangerous**. If we don't figure out how to properly compensate and recognize maintainers for the value they create, we might wake up one day and find that the projects we rely upon most are no longer being maintained at all. ■

HEADLINE #9

In the wake of the xz utils hack, two-thirds of maintainers are less trusting of contributors

In late March 2024, a developer from Microsoft noticed some unusual behavior on their computer, investigated it, and uncovered a hack of epic scope in an obscure but important library called xz utils. The attack was technically sophisticated, but perhaps worse, it was socially sophisticated. The attackers took advantage of an open source maintainer over a long period of time (years) to slowly, but steadily, win his trust—and then subvert the security mechanisms that he had previously put in place.

The maintainer facing this deliberate, long-term attack was, in his own words at the time the hack began, “unpaid.”

> “I haven’t lost interest but my ability to care has been fairly limited… it’s also good to keep in mind that this is an unpaid hobby project.”

In the same email, this maintainer said:

> “Jia Tan may have a bigger role in the project in the future. He has been helping a lot off-list and is practically a co-maintainer already. :-)”

It was exactly this “Jia Tan” who, over a period of two years, took over xz and inserted a malicious backdoor that could have exposed computers the world over to remote execution. Thankfully this attack was discovered before it could cause extensive damage.

But we wanted to use this year’s maintainer survey to find out whether the xz utils hack has inflicted any collateral damage; namely, has it negatively impacted the way open source maintainers think about their work?

First, we got a baseline read on how many maintainers were even aware of the xz utils hack. We provided a brief description of the hack, and then asked them if they were aware of the hack prior to taking the survey. The vast majority of maintainers (88%) were already aware.

How has the xz utils hack impacted maintainer trust?

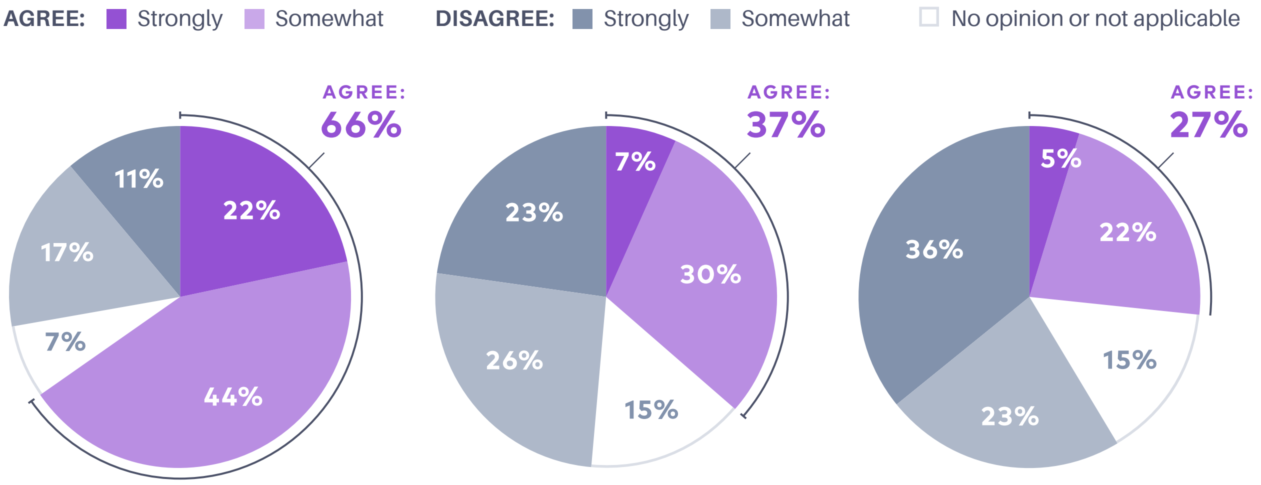
Next, we asked those that were aware of the xz utils hack the following question:

> How much do you agree with the following statements in terms of how the xz utils hack has impacted the way you approach your work as an open source maintainer?
> - I'm less trusting of the contributions of my co-maintainers or feel I need to vet them more carefully
> - I'm less trusting of pull requests from non-maintainer contributors or feel I need to vet them more carefully
> - It has added to my personal stress

Of the three statements, maintainers most agreed with: "I'm less trusting of pull requests from non-maintainer contributors or I feel I need to vet them more carefully." Two-thirds (66%) either agreed or strongly agreed with that statement, which shows that the xz utils hack has had a significant impact on maintainer trust of contributions.

Two-thirds of maintainers aware of the xz utils hack are now less trusting of pull requests from non-maintainers

Respondents aware of the xz utils hack were asked, "How much do you agree with the following statements in terms of how the xz utils hack has impacted the way you approach your work as an open source maintainer?"



I'm less trusting of pull requests from non-maintainer contributors or feel I need to vet them more carefully

I'm less trusting of the contributions of my co-maintainers or feel I need to vet them more carefully

It has added to my personal stress

n=307. Totals may not equal 100% because of rounding.

We also asked maintainers to tell us in their own words how the xz utils hack has impacted their work, and several spoke directly to this point.

> “I'm definitely taking code review more seriously and no longer merge code I don't understand. Worst case, the author will have to walk me through line by line.”

Or as another maintainer put it:

> “It made me more aware of the sophistication of actors looking to compromise projects, and made me more mindful of vetting non-trivial code changes.”

But that degradation of trust also creates challenges when it comes to encouraging contributions, as another maintainer observed:

> “I feel the need to add a layer of vetting, but adding any additional layer of friction to a possible open source contributor would just scare them away. I cannot afford to be pushing people away.”

Thankfully, on the other two statements we asked about, the results were less concerning. Only 37% agreed or strongly agreed with the statement:

> “I’m less trusting of the contributions of my co-maintainers or feel I need to vet them more carefully.”

One maintainer felt like xz had brought them closer:

> “Made me appreciate my co-maintainer.”

Another felt that even though the time horizon on this hack was long, their co-maintainer relationships had proven trustworthy over an even longer timespan:

> “While the xz hack was longer than might be expected, the time spent by my co-maintainers is still longer than that, so I feel confident in their reliability.”

And an even smaller percentage (27%) agreed or strongly agreed with the statement “[xz utils] has added to my personal stress.” One maintainer who agreed with that statement made the following observation:

> “Now, checking in code from other contributors is stressful."

Maintainers express the impact of the xz utils hack in their own words

Overall, by analyzing the open text responses, we were able to categorize maintainers' concerns expressed in their own words into a few high-level categories. The most-often-shared sentiment was that, in the wake of the xz utils hack, maintainers will need to increase their caution during code review and vetting.

One maintainer expressed concern that even if contributions were technically sound, that might no longer be enough.

> "Now I have to be extra careful with reviewing pull requests from some random GitHub user. Even their proficiency doesn't make you believe them."

Several other maintainers indicated they'd be taking a more cautious approach in the future, for example:

> "I am MUCH more suspicious of any code contributions from non-maintainers, especially from individuals who have not been active in our user community."

Some maintainers were concerned about what the xz utils hack says about how we can rebuild trust and improve community dynamics.

> "This incident really highlighted for me that technology is not the problem—culture is. Without authentic, trustworthy support from a real community (not merely an accidental collection of strangers who have a single common interest) this kind of thing will only continue. Security is a wetware problem first and foremost—we need to care about actual, living humans, not just certs and hashes and chains of custody."

> "It's not a new problem, this issue has always been there in some shape or form, it's just shined more light on an existing problem (which is good). But we still need actionable things that can be done to help mitigate these problems. Ring of trust, reproducible builds, etc. lots of these things are great ideas, but they are all half implemented. (e.g. how many companies actually validate GPG signatures from packages they download, with an existing ring of trust, reproducible builds are great, but they don't provide a mechanism to certify the rebuild process)."

And, notably, many maintainers felt that the hack would have no significant impact on their work at all. As one put it:

> “Trusting new maintainers by default is the open source way, and how it should remain. Throwing everyone under the bus just because of one threat actor is not a good idea. I added a new maintainer to a security-critical package I maintain recently, and I vetted them as I would have vetted them before xz utils—see past contributions, and keep an eye on all the changes they are pushing.”

Finally two maintainers nicely summed up the entire discussion with these thoughtful words:

> “I’ve always been aware of the possibility of a sabotage contribution. I’ve refreshed my thinking on the risk and doubled down on the need to validate each solution for its merits and not accept a change that I wouldn’t be happy putting my own name on the commits.”

> “I think the case has, despite clearly being a bad thing, helped shine a brighter light on the whole supply chain and how we need to have clear and open standards for every aspect of projects, not just code or documentation.”

Well said. ■

HEADLINE #10

AI-based coding tools are thriving, and maintainers have some valid concerns about the impact on their work

Nothing says it is 2024 in the technology industry like a headline about AI, so we would have been remiss if we *hadn't* asked maintainers a few questions about their perceptions of AI. One of the biggest AI-related headlines of 2024 has been the rapid growth and acceptance of AI-based coding tools. So we wanted to start by asking maintainers to share their assessment of the impact that AI-based coding tools will have on their maintenance work.

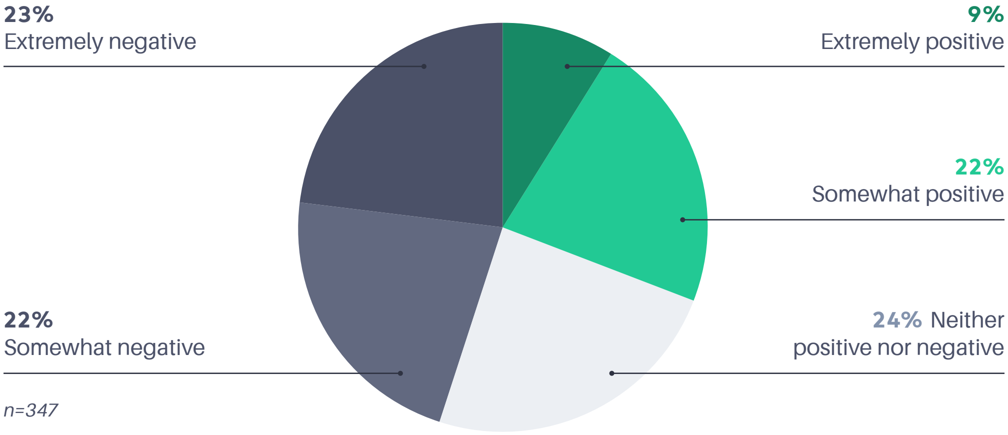
What do maintainers think will be the impact of AI-based coding tools on their work?

First, we got a baseline impression of whether the impact of AI-based coding tools on maintainer work was positive or negative. The overall maintainer perception of AI-based coding tools leaned negative, with almost half (45%) of maintainers predicting that these tools will have a somewhat negative (22%) or extremely negative (23%) impact on their work.

In fact, maintainers split pretty evenly across every choice but "extremely positive" (9%), with 22% of maintainers saying AI-based coding tools will have a somewhat positive impact and 24% saying they will have neither a positive nor a negative impact.

Maintainers' perceptions of the impact of AI-based coding tools on their work is varied, but leans negative

What impact do you believe AI-based coding tools will have on your work as an open source maintainer?



n=347

Maintainers explain the impact of AI-based coding tools in their own words

We gave maintainers a chance to elaborate on their response to the previous question. After categorizing the comments, we found that the highest percentage of responses voiced concerns about the quality of code AI-based coding tools currently produce.

As one maintainer said:

> “My experience is also that AI-based tools often produce incorrect code in more complex situations, and it can be hard to identify issues with their code unless you already know how to do it.”

Other maintainers agreed with that assessment, and added more context:

> “AI-based coding tools exponentially increase the chance that someone without context of the codebase or the project will build a PR that looks correct but contains breakages that can bring the entire language ecosystem down until a patch fix is released. The codebase is simple but the impact is absolutely beyond enormous.”

> “AI makes it easy to quickly generate lots of code which nobody understands, including the AI creator. As a technology, it is great at simple stand-alone tasks, or boilerplate which aligns well with existing code upon which it has been trained. As such, it is an occasionally useful tool for working programmers. But used for anything deeper, it frequently generates code with errors both subtle and glaring, and has to be carefully and fastidiously corralled into the desired behavior by an expert level programmer.”

The next most common concern voiced by maintainers was the increased maintenance burden they believe AI-based coding tools will create for them.

A few example comments:

> “The increase of spam PRs, comments, and false positives from AI tools and users has been enormous and very frustrating.”

> “LLMs and machine learning tools have demonstrated potential aid only for the mechanical, non-creative aspects of software development. These are at the expense of increased burden to vet their output for mistakes, and the tools are incapable of explaining their work, so this is worse than with a human collaborator.”

> “I don’t want to become the gate for reviewing tons of automatically generated pull requests. Sounds like it would further wear me and my co-maintainers down.”

Other maintainers’ comments expressed critiques of the current generation of AI-based coding tools.

> “I don’t find AI-based coding tools useful yet, there’s a lot of nonsense in the suggestions and they don’t feel well-integrated into coding tools yet. For example, they may override or be confused for type-based suggestions. I imagine those issues will be fixed, and these tools will save some time with boilerplate tasks, but also introduce overhead elsewhere. I don’t expect a huge impact on my life either way.”

> “I don’t mind AI making suggestions, but all suggestions whether by an AI or a human require checking and thought. First and foremost programming towards a specific goal requires a clear understanding of the problem and clear thinking about the ways to best accomplish this. Sometimes deeper theorems about the problem and special algorithms are needed. For these kinds of things AI has not been very helpful. I am not even sure AI purports to do this kind of thing either.”

Still, despite the reservations, a good number of maintainers expressed optimism about the possibilities of AI-based coding tools.

> “I use Copilot and I’m sure it will take adjusting to get used to these new tools but I think the payoff is definitely going to be worth it. They just need to be used within reason.”

> “I am hopeful that it will help me with the boring tasks I keep putting off, such as documentation and code tests.”

> “For me GitHub Copilot has been a spark that has given me interest in maintaining my projects again because it cuts a lot of the mundane parts out.”

How willing are maintainers to review and accept contributions created using AI?

We also wanted to learn more about how willing maintainers would be to accept code contributions that they knew were produced using AI-based coding tools. We asked:

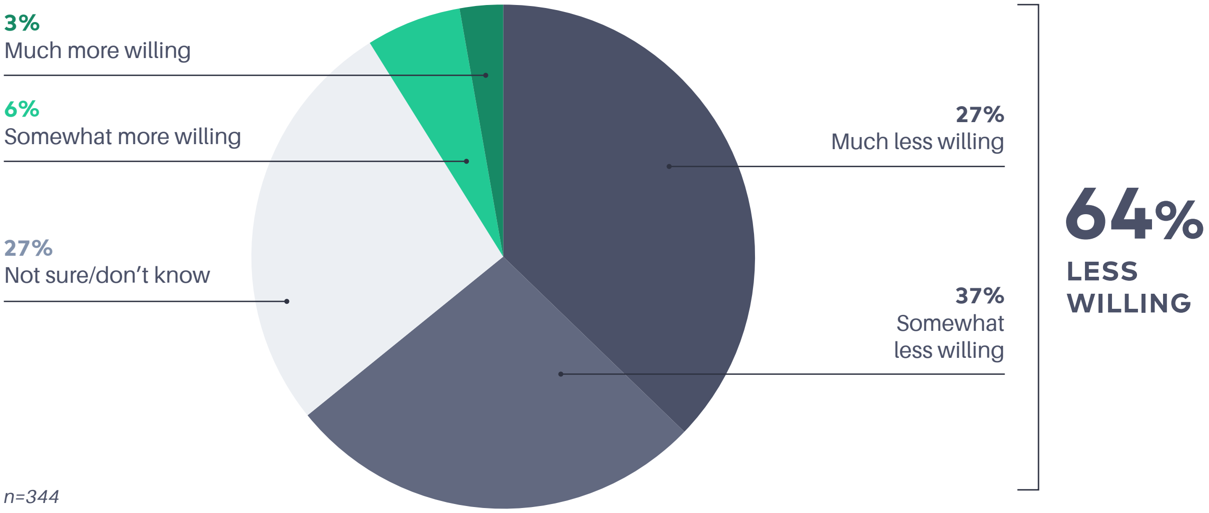
> If you knew code contributors were using AI-based coding tools, how would this impact your willingness to review and accept their contributions?

On this question, maintainers' responses were significantly more negative than their general perceptions of AI-based coding tools. Almost two-thirds of maintainers (64%) said they'd be much less willing (37%) or somewhat less willing (27%) to review and accept contributions they knew were produced using AI-based coding tools.

Only 9% said they would be much more willing (3%) or somewhat more willing (6%) to review and accept contributions they knew were produced using AI-based coding tools. And just over one-fourth (27%) aren't sure or don't yet know enough to make a decision.

Almost two-thirds of maintainers would be less likely to review and accept contributions created using AI-based coding tools

If you knew code contributors were using AI-based coding tools, how would this impact your willingness to review and accept their contributions?



n=344

How useful is the information from automated pull requests for vulnerability remediation?

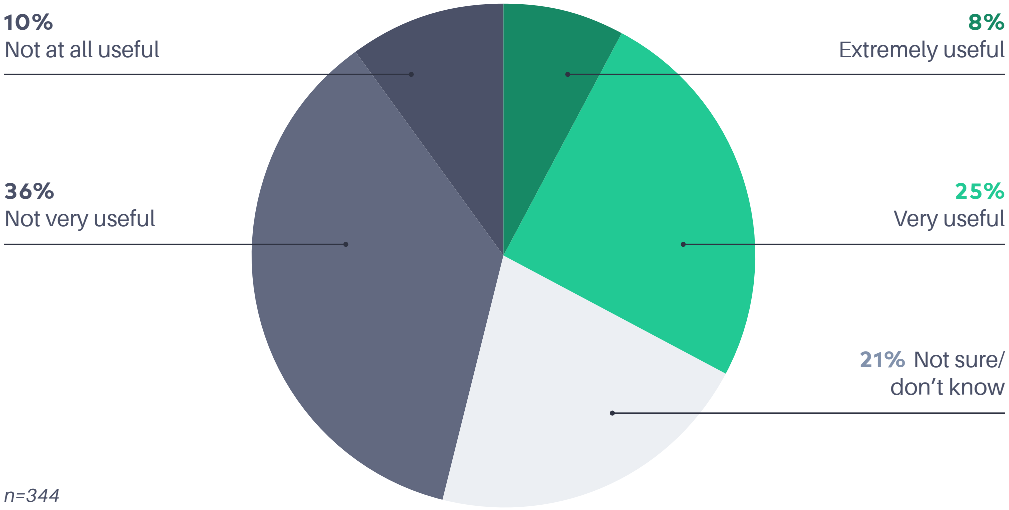
Finally, because we've heard reports that maintainers were receiving many more AI-generated pull requests for vulnerability remediation than they have in the past, we wanted to get a sense for how useful these pull requests are.

On this question, the bulk of maintainers responded toward the middle of the spectrum and not at either extreme. The highest percentage of maintainers (36%) find the information they receive from automated pull requests "not very useful" for vulnerability remediation. The next most popular response was "very useful" (25%), with only 10% at the negative extreme answering "not at all useful" and 8% at the positive extreme answering "extremely useful."

As with some of our other AI-related questions, a good percentage of respondents had not yet made up their minds: 21% answered that they are not sure or don't know whether the information they receive from automated pull requests are useful for vulnerability remediation.

For many maintainers, the jury is still out when it comes to the usefulness of information from automated pull requests for vulnerability information

In your experience, how useful is the information you receive from automated pull requests for vulnerability remediation?



n=344

It’s a pretty clear bet that AI-based coding tools are here to stay. And in our survey maintainers raised some valid concerns regarding how these tools will impact their maintenance work, although many also see a lot of positive potential in AI-based coding tools as well.

To best serve the needs of open source maintainers, the ideal path the creators of AI-based coding tools will need to navigate as they continue to innovate is to ensure they *remove* more maintainer work than they create and, at least with maintainers, on that front there are many good suggestions in this survey for making this a reality. 🟩

HEADLINE #11

Younger open source maintainers are significantly more likely to use AI-based coding tools

In our previous finding, we learned quite a bit about open source maintainers' perceptions of AI-based coding tools and how these tools are impacting their work as an open source maintainer today. But we didn't stop there. We also thought it would be interesting to better understand how maintainers are using AI in their own work.

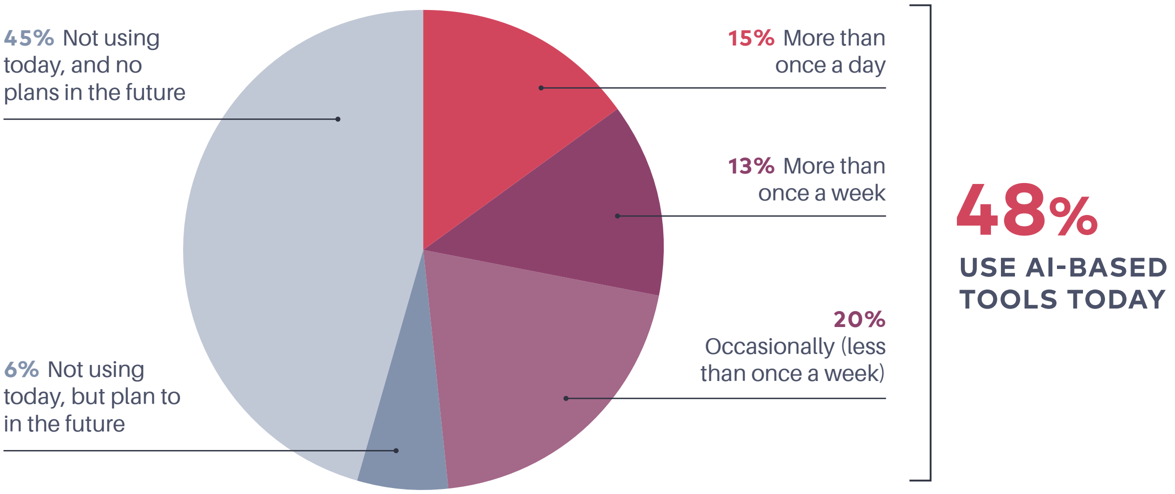
Are maintainers using AI-based coding tools today?

We asked maintainers to tell us how often they are using AI-based coding tools in their work today. About one-half (48%) are using these tools already, with 20% using them occasionally (less than once per week), 13% using them more than once a week, and 15% using them more than once a day.

Those who are not using AI-based coding tools mostly have no plans to use them in the future either, with 45% selecting that option and only 6% not using them today, but planning to in the future.

How often open source maintainers use AI-based coding tools

Which of the following best describes how often you use AI-based coding tools for your work as an open source maintainer today?



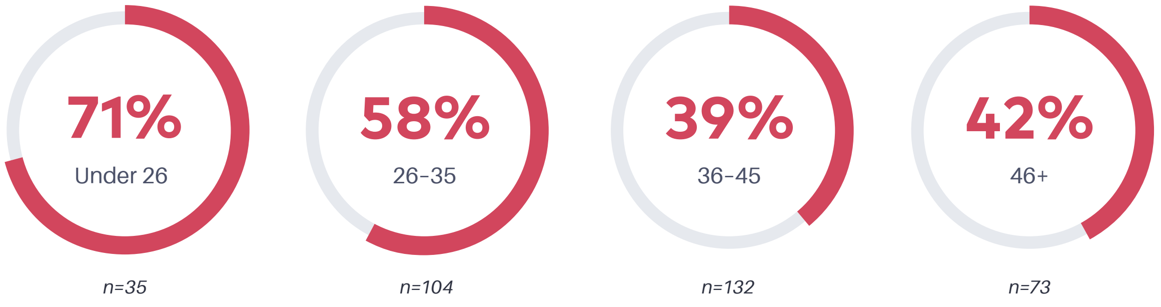
n=345

Interestingly, use of AI-based coding tools is highly correlated with the age of the maintainer. Splitting the same data by age, we find that 71% of maintainers under age 26 are using these tools at least occasionally (+23% over the full sample).

This percentage drops to 58% for maintainers between 26–35 years old, and then to 39% for maintainers 36–45 years old, and finally increasing slightly to 42% for maintainers 46 years old or older.

Younger open source maintainers are significantly more likely to use AI-based coding tools

% of those using AI-based tools at least occasionally



How maintainers are using AI-based coding tools, in their own words

We asked those maintainers who are currently using AI-based coding tools to tell us in their own words what they were using the tools for. The highest percentage of respondents said they were using AI-based coding tools for code completion and suggestions, and the vast majority of these respondents are using either GitHub Copilot or ChatGPT.

A few example comments:

> "I use Copilot pretty much every day for very basic tasks like remembering how to implement common traits like std::fmt::Display. I don't use Copilot for the overall structure, algorithms, or architecture."

> "I use Copilot for auto-completion of code, useful for the boilerplate I like to establish in my projects. I also use a custom discord bot that leverages GPT-4o for brainstorming and light delegation of simpler coding tasks."

> "I use ChatGPT for my day-to-day coding. I may ask it to help write new features, write CSS, diagnose errors, or generate boilerplate code to start a new feature/project."

Many respondents report using AI-based coding tools, but do so with some concerns. For example:

> "I use GitHub Copilot integrated with VS Code and RStudio. I occasionally use the code it suggests, but I nearly always have to modify it to make it correct."

> "They still make mistakes and can be dumb, but I use them daily for ideas, chore work like generating tests and types, etc. It has made me more productive as an open source maintainer."

> "While I know first-hand that AI can act as a good 'research assistant' especially for newcomers—full of ideas but perhaps not full of the required skills yet—I have yet to see AI be sufficiently helpful in matters where context is key, as is often the case in the more nitty gritty parts of open source projects."

Another common way maintainers use AI-based coding tools is for documentation and testing.

> "I'll often use AI to help build test cases, or to help me refactor difficult-to-understand code."

> "I use them sometimes to write tests. I can show it existing tests and ask it to write some new ones based on those."

> "I have started recently using it to provide better and more detailed commit messages. Those commit messages will end up in the changelog when releasing a new version. So it can help provide a more detailed changelog in the future."

What kinds of open source software-related problems could be solved by AI?

We ended the AI section of the survey by asking maintainers to share with us any final thoughts about the types of open source problems that might be solved by AI.

We were able to tease out quite a few interesting ideas from maintainers for how AI could help improve open source. Here are the main categories of ideas, along with a few example quotes for each.

Documentation: help improving documentation, automating documentation tasks, and making documentation more accessible.

> “Non-technical problems like changelog summaries or other similar boring tasks about presenting the content of technical actions to lay people. Perhaps some documentation related text, auto-extracted from the source code.”

> “Creating release notes…. Automatically turn README files and Wiki entries into a chatbot.”

> “GitHub Community questions could have an ‘AI’ proposed solution given to me that I can approve if it’s correct to show as an answer. This could speed up answering similar questions multiple times with similar code.”

Issue triage: help automating issue triage, identifying duplicate issues, and prioritizing issues.

> “Sometimes I receive vague bug reports or feature requests. I think having a chatbot that assists reporters and contributors in creating these could help reduce such cases.”

> “Ensure issues have all the necessary context. Provide first answers for queries (especially for first-time users), with the context of the docs and tests. Attempt to fix trivial issues when people submit PRs (e.g. lint errors, breaking changes in Dependabot version updates)—most fixes are trivial, especially when looking at error messages and (in case of version upgrades) changelogs.”

Code quality and review: help automating code review and improving code quality.

> “Resolve imports to dependencies needed to satisfy those imports. Provide intelligent refactoring. Assess safety of a given change. Generate tests and PRs to capture and resolve a reported issue.”

> “Better fuzzing to detect and perhaps even auto-fix classes of bugs. For example, CPython reference counting issues.”

Dependency management and security: help automating dependency management, identifying security vulnerabilities, and updating dependencies.

> “Given a changelog for a new release of one of my dependencies, and the way the dependency is actually used in my codebase, what changes in the dependency do I need to investigate further than my tests will cover?”

> “Filter vulnerability reports so they are about dependencies my project actually uses and not just part of the build tools.”

And with that, we end our tour of maintainers’ thoughts and perceptions of AI. We now have a good sense for what maintainers think about AI-based coding tools and how they expect these tools will impact their work. We also learned more about how many maintainers are using AI-based coding tools today, and what they are using them to accomplish. And we ended by getting some specific ideas from maintainers for how they think AI could be used to solve open source software related problems. ■

HEADLINE #12

The open source maintainer community is getting grayer

Since we've now fielded this maintainer survey three times in the past four years, we thought it might be interesting to look at some of the overall demographics of the maintainer community, and see if any of these demographics have changed over time.

Is the maintainer population growing older?

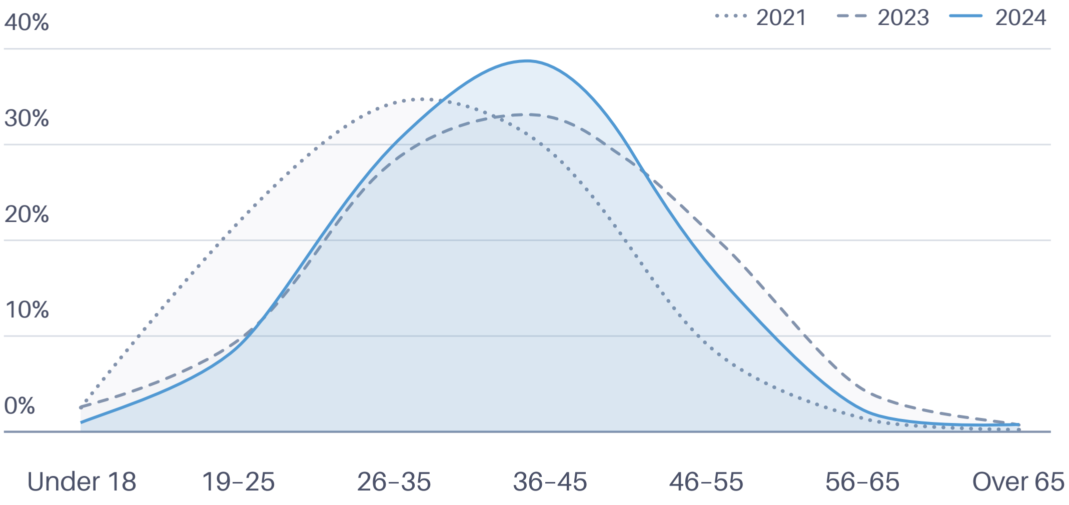
Because we've seen several articles recently covering the "graying" of open source, we figured with three years of data under our belt, we'd be uniquely positioned to provide a data point to answer whether the open source maintainer community is aging.

And to cut to the punchline, yes, open source maintainers are getting older.

We plotted out the ages of open source maintainers from the first survey we completed in 2021 through this year's survey, and what it shows us is that the percentage of maintainers self-reporting that they are 46–55 or 56–65 has doubled since our first survey in 2021 (2021: 11%; 2023: 27%; 2024: 21%). Meanwhile, the percentage of maintainers under 26 has dropped precipitously from 25% in our 2021 survey to 12% last year and 10% today.

The open source maintainer community is getting grayer

How old are you?



n=344 (2024); n=259 (2023); n=322 (2021)

It would be easy to speculate reasons why the current maintainer population is aging, and new, younger maintainers aren't coming in to fill the gaps. Perhaps many of the things we've learned during the course of this survey report are making being an open source maintainer a less appealing hobby or profession.

After all, we learned that almost half of maintainers feel underappreciated and like the work is thankless, and many also feel like it adds to their stress and that they are not financially compensated for the work. Almost two-thirds of maintainers have quit or considered quitting their maintenance work.

It is also possible that existing maintainers who have not quit are continuing to stick with their projects, and the demographic change is simply a result of the age of the open source movement overall, where many maintainers are getting older, but not yet of retirement age, and perhaps we'll see a youth movement again as they hand over the reins to a new generation of maintainers in the coming years.

Or maybe there is an unsolved challenge related to training new maintainers that needs to be addressed. Perhaps we need more formal mentorship or skills-based-training programs to teach maintainers the necessary skills, especially as the job gets more complex and demands from enterprise users and governments continue to grow.

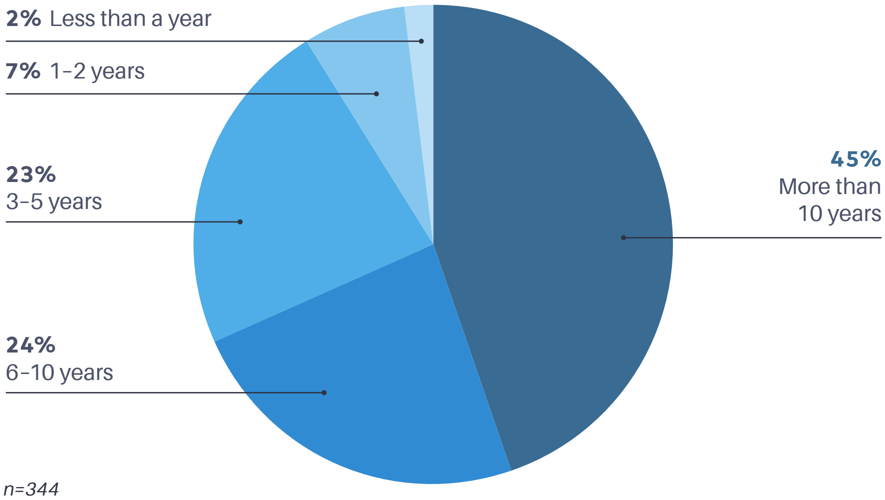
Other demographic information about the maintainer community

In addition to age, for the first time this year, we asked maintainers how long they've been an open source maintainer. Almost half of respondents (45%) have been open source maintainers for more than 10 years. Meanwhile 24% have been maintainers for 6–10 years and 23% have been maintainers for 2–5 years.

Only 7% of respondents reported that they've been a maintainer for 1–2 years and 2% reported that they've been a maintainer for less than a year (see chart on the following page), which may be another troubling signal that the current crop of maintainers is aging and not being replaced by a new generation.

Almost half of maintainers have been doing the work for more than 10 years

How long have you been an open source maintainer?



n=344

We asked about maintainer gender for all three years, and the data there has been mostly consistent. The vast majority of maintainers (85%) identify as male, which is similar to previous years (85% in 2021 and 83% in 2023).

Only 6% of maintainers identify as female, which is slightly down from 8% in 2021 and 9% in 2023, but probably not statistically significant given the sample size. The percentage of maintainers identifying as non-binary has increased from 1% in 2021 to 2% in 2023 to 3% this year, but again, the sample size is small enough that it would be hard to read much into the data accurately. And 6% of maintainers prefer not to share their gender, which has been roughly flat over the years (6% in 2021 and 5% in 2023).

Maintainer gender, 2021-2024

What is your gender?

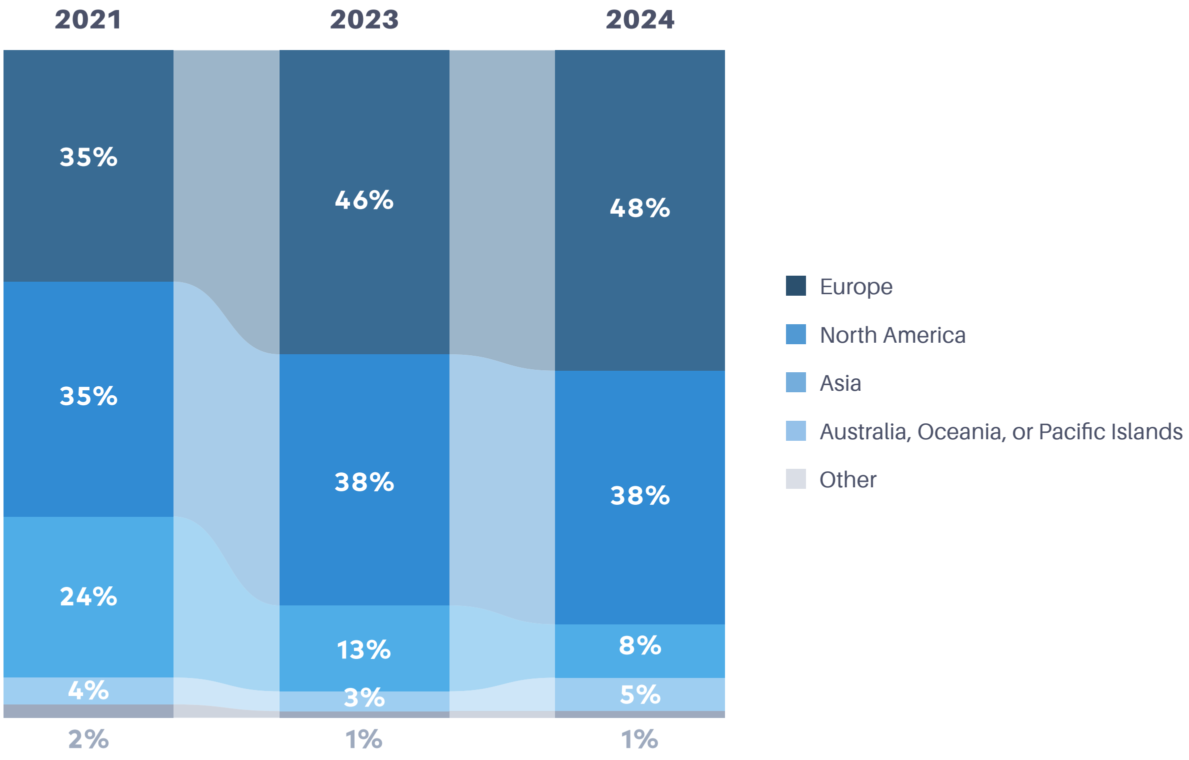
	2021	2023	2024
Man	85%	83%	85%
Woman	8%	9%	6%
Non-binary	1%	2%	3%
Let me type	0%	1%	0%
Prefer not to say	6%	5%	6%

n=344 (2024); n=259 (2023); n=322 (2021)

Finally, we once again asked maintainers where in the world they live. One caveat when looking at this data is that we offered to send a free t-shirt to maintainers in North America, South America, Europe, and Australia who filled out the survey 2023 or 2024, while in 2021 we sent t-shirts to maintainers anywhere in the world. This helped skew the 2021 numbers heavily toward Asia.

Geographic distribution of open source maintainers

What geographic region are you located in?



n=344 (2024); n=259 (2023); n=322 (2021)

So this data is helpful for contextualizing the survey data we've collected over the years, but may not be representative of the worldwide distribution of open source maintainers.

With that in mind, it is still interesting to see that European countries represent the largest group of open source maintainers (48%), followed by North American maintainers (38%), Asian maintainers (8%), and those from Australia, Oceania, or the Pacific Islands (5%). ■

Look at that! You've made it to the end.

And with that, we've reached the end of the 2024 state of the open source maintainer report. If you've stuck with it all the way until you are reading this, congratulations, and thank you for your interest in the subject!

We hope this has been a good use of your time. If you agree, please share the report with others who you think might find it useful as well. If you have questions about the data in the report, find any errors (gasp!), or would otherwise like to discuss the report findings with us, we'd love to hear from you. Email press@tidelift.com with the subject line "2024 Tidelift state of the open source maintainer report" and we'll route your email to the best people to reply.

Thank you for caring about the state of open source maintainers!
They (and we) appreciate you!

Acknowledgements

It takes a team to produce a report of this size, and we want to give special thanks to Lawrence Hecht for his work on survey design, programming, and analysis, plus being an all-around great brainstorming partner and helping wrangle the data to uncover the most interesting outcomes. Thanks also to Tatiana Temple for her excellent, detailed, and creative chart design work.

Many people on the Tidelift team contributed to the report including survey design, writing, project management, editing, and general cat herding. Thanks in particular to Chris Grams, Kanish Sharma, Amy Hays, Caitlin Bixby, Lauren Hanford, Luis Villa, Donald Fischer, Jeremy Katz, Havoc Pennington, Jeremy Rissi, and others who helped along the way.

Finally thanks to the more than 400 open source maintainers who took the time to share their thoughts and make this a useful reference for the state of being an open source maintainer in 2024.

TIDELIFT.COM

TIDELIFT