THE 2024 TIDELIFT MAINTAINER IMPACT REPORT

November 2024

TIDELIFT

Introduction

Open source is everywhere. Almost all modern applications contain open source, and in many applications it makes up 70% or more of the code. For good reason: open source speeds up innovation, and it is almost impossible to imagine a world today where applications are developed without it.

But as open source continues to become more ubiquitous each year in enterprise applications, the risks organizations using open source are expected to manage also grow. High profile security vulnerabilities impacting open source, headlined by the Log4Shell and xz utils incidents, have made open source software security a critical board-level issue in many organizations, potentially impacting revenue, data, and customers.

Governments around the world are stepping up efforts to improve cybersecurity and reduce risk to consumer data and national security. Efforts like the Cyber Resilience Act (CRA) and Product Liability Directive (PLD) in the EU and the National Cybersecurity Strategy and White House Executive Order 14028 on Improving the Nation's Cybersecurity in the United States place new responsibilities on organizations building software applications to ensure their products are secure, and codify paths to pursue damages if they are not. Many organizations already have efforts underway to ensure the software they build in-house complies with these new regulations, but it is still an open question for many how they will keep the open source code that in many cases makes up the majority of their application secure and compliant as well.

Whose job is it to keep open source secure?

So whose job, exactly, is it to ensure the open source code that most modern applications rely on is secure, well maintained, and resilient?

Many people believe the security and maintenance of open source code is the job of the maintainers who write it, and commercial users do not bear the responsibility for insecure open source software they bring into their applications.

This perception is not only wrong; it is dangerous.

As we'll share in more detail in this report, much of the open source software our enterprises rely on is built and maintained by **volunteer maintainers who aren't being paid to keep it up to date and secure**. Sixty percent of maintainers report they are unpaid hobbyists, and only 12% of maintainers consider themselves professionals who earn most or all of their income from their maintenance work.

Open source code downloaded from the internet is being provided "as is," which means the user—**your organization**—is taking responsibility for any issues that arise from its use.

This responsibility is being codified into new cybersecurity regulations like the aforementioned Product Liability Directive in the EU. Here's a direct quote from the PLD (emphasis ours):

> "Where free and open source software supplied outside the course of a commercial activity is subsequently integrated by a manufacturer as a component into a product in the course of a commercial activity and is thereby placed on the market, **it should be liable to hold that manufacturer liable for damage caused by the defectiveness of such software** but not the manufacturer of the software because they would not have fulfilled the conditions of placing a product or component on the market."

In other words, if your organization makes the decision to use an open source component that is later found to be insecure and it results in damages, **your organization would be liable for the damages**—not the open source maintainer who created the component.

The impact of paying open source maintainers to keep their projects secure

Open source is an amazing resource, and it is not a realistic option for organizations to quit using open source and expect to stay relevant. A recent Harvard Business School study estimated that it would cost 3.5 times more to build software without utilizing open source. Most organizations can't wait 3.5 times as long or spend 3.5 times as much to build their applications and still remain competitive in their industry. They need a way to take full advantage of open source while reducing risks to their revenue, data, and customers.

Thankfully there is an answer for how organizations can continue to get the business benefits of open source while still controlling their security risk: **paying maintainers to do the important work to ensure their projects follow secure software development practices—and continue to do so into the future**.

We wrote this maintainer impact report to share the evidence of the positive security outcomes that organizations can expect to achieve—outcomes that reduce organizational risk and improve operational efficiency—when they invest directly in their open source software supply chain by paying maintainers.

We'll share this story in two parts. First we'll share a case story of how one customer improved the security and resilience of an important Python application used to analyze and forecast commercial pricing in a highly regulated industry over a multi-year period while simultaneously saving over $1 million in organizational time and significantly reducing application risk.

Next, we'll share recent evidence from our 2024 Tidelift state of the open source maintainer report where maintainers report first hand what kinds of security practices they are able to implement when they are paid for their work. We'll also bring in supporting data from other sources, like the Sonatype State of the Software Supply Chain report and the Atlantic Council report O$$ Security: Does More Money for Open Source Software Mean Better Security?

We'll end by sharing how Tidelift can help organizations use these findings to decrease their own risk and improve organizational efficiency proactively, while still making optimal use of open source. ■

PART 1

Case story: the business impact of paying open source maintainers to scale real world application security

What if your team could save $1 million while improving the security and resilience of an application that your company depends on to deliver revenue? Here's a story about a team that did just that.

As we stated in the introduction, over the past few years organizations have become increasingly focused on improving the security of the open source powering their applications. Vulnerabilities like Log4Shell in 2021 have highlighted the risk to organizations' revenue, data, and customers that can result from insecure or under-maintained open source packages.

Some industry leaders have pushed for codifying security practices for open source projects, and then measuring how well projects follow these practices. The OpenSSF Scorecard has been the most visible of these efforts. But while a Scorecard score may provide good information about the security, health, and resilience of a package when the data is accurate, **it is not actually making the package more secure, healthy, or resilient**. The Scorecard is simply a historic snapshot of observable activities.

For this maintainer impact report, we wanted to analyze the impact of paying open source maintainers to implement industry-leading secure software development practices in real world situations, and see if it is making a difference in customers' bottom line and their ability to increase the velocity of their business.

As a case story, we zeroed in on a Python application that one of our customers uses to analyze and forecast commercial pricing in a competitive, highly regulated industry. We wanted to see how they were able to improve the application's security and resilience over a two-year period with help from Tidelift and our open source maintainer partners.

Let's start with the bottom line results: This customer is able to set accurate pricing, drive profitability, and improve their margins because their developers have been able to reduce the organization's reliance on abandoned, end-of-life, or otherwise insecure open source packages that are costing them time and money.

Specifically, they:

- Saved $1.1 million of organizational time across engineering, legal, and security that would have been spent on requirements research and engineering implementation time
- Reduced application risk by turning 37% of this customer's independently maintained packages from an "unknown future" to reliably secured and maintained, with a plan in place to grow that percentage to 58% in 2025 and 80% in 2026

All of this return is in addition to the organization's ability to continue to take advantage of the increased business velocity that open source provides.

Improving visibility

In 2021, before the customer began to track this application with Tidelift, they had limited visibility into the health, security, and resilience of their open source dependencies. At best, the customer would have visibility into CVEs affecting package releases in the application, alongside information from the OpenSSF Scorecard about package maintenance and discoverable security policies.

After starting to work with Tidelift, their visibility into the health, security, and resilience of their open source software supply chain increased radically. On day one, the customer could answer questions like:

- What are the known CVEs affecting package releases?
- Which of these CVEs are false positives, or don't affect the way the package is used?
- Which packages have a maintainer available to take in security research? Is security research being ignored?
- Which packages are backed by foundations or corporations?
- Does the package maintainer release patches for known vulnerabilities?
- Which packages have been abandoned?
- What is the abandonment risk from transitive open source packages pulled in by direct dependencies?
- What packages are showing signals that they may go "end-of-life" or be abandoned?

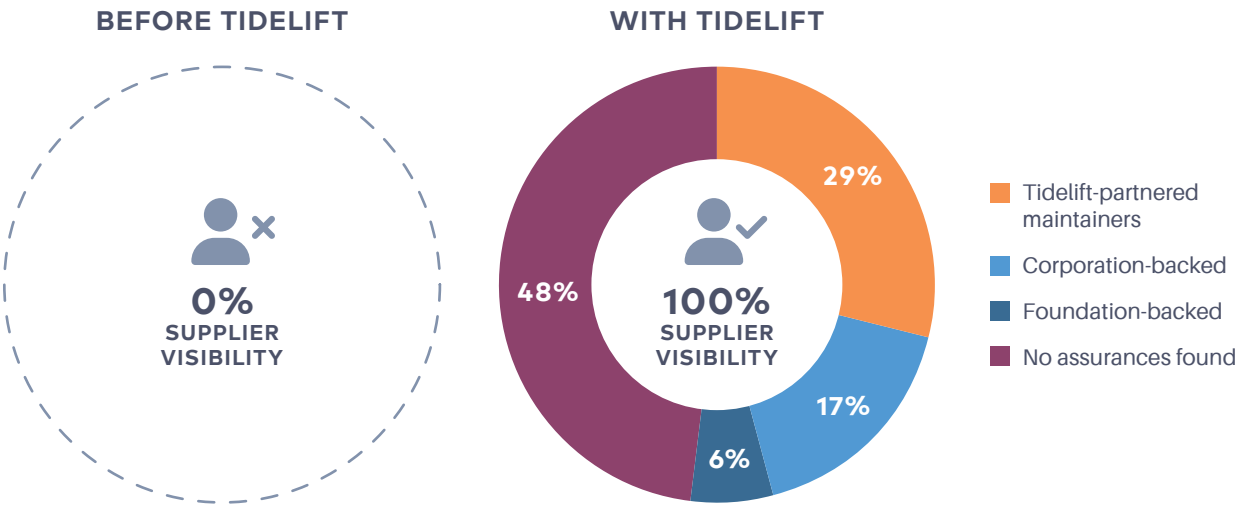
This customer immediately had much more visibility into which packages were known to be following secure software development practices, and which were not. And, since all open source packages are not created equally, with some backed by independent open source maintainers, and some by large corporations or foundations, they were immediately able to see the benefits of Tidelift's partnerships with independent open source maintainers.

100% supplier visibility

The chart below shows how the organization went from having zero visibility into who was maintaining their software, and the secure development practices behind those open source dependencies, to 100% supplier visibility. This new visibility into supplier information, accurate health and security data, and long term resilience reduced the overwhelm of an endless list of CVEs. The customer now had actionable information, and a way to proactively ensure that the packages they needed to keep in place would be reliable.

Open source supplier visibility before and with Tidelift

Before they started working with Tidelift, this customer had no visibility into who their open source software suppliers are, or to what degree those suppliers are following secure development industry standards.



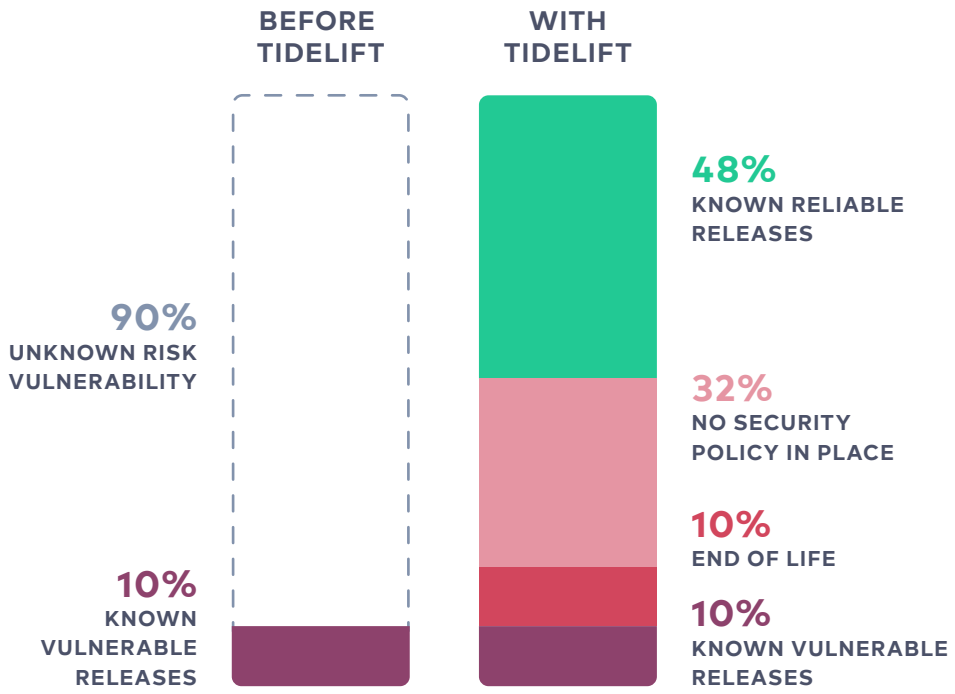
The company instantly had contractual commitments from the maintainers of 19% of their dependencies who were already being paid by Tidelift to implement enterprise-grade secure software development practices, and continue to keep these practices in place over time. Over the following two years, Tidelift recruited additional maintainers, and now the percentage of packages backed by Tidelift-partnered maintainers is at 29% and continuing to rise. In addition, the customer also now knew which packages were backed by independent open source maintainers who were not paid by Tidelift to follow secure software development practices and didn't have the safety net of affiliation with corporations or foundations (the 48% we refer to as "no assurances found" in the chart above).

100% risk visibility

Meanwhile, before beginning work with Tidelift, the customer had some risk visibility via their software composition analysis (SCA) tool, which could show them which package releases in their application had known vulnerabilities. But known security vulnerabilities are only one indicator of potential risk, and they had no visibility into other risk vectors, including from packages that had been declared end-of-life or did not have a security policy in place.

With Tidelift, they now have 100% risk visibility, with data on other forms of risk beyond security vulnerabilities. Almost half of the packages in the application (48%) are now known to be reliable (with no known security vulnerabilities, a discoverable security policy available, and no indication that they've been declared end of life). Meanwhile, in addition to the 10% of known vulnerable releases, they can also see that 10% of their package versions have been declared end of life, and 32% have no security policy in place. **End-of-life packages and packages with no security process are both important risk vectors that are invisible to traditional SCA tools**.

Open source risk visibility before and with Tidelift



Note: A package is considered "known reliable" if it has no known security vulnerabilities, a discoverable security policy available, and hasn't been declared end-of-life.

With help from Tidelift, the customer went from having limited visibility into who their open source suppliers were and limited visibility into their open source risk to having full visibility into both their suppliers and their risk.

They now had contractual relationships with the independent maintainers of 29% of their open source dependencies, and data showing foundations or corporations backing another 23% of their dependencies.

Increasing supplier and risk visibility was a good start. But in order to have the largest possible impact on this organization's ability to use open source effectively, Tidelift's mission now became "how are we going to improve the security and resilience of the other 48% of independently maintained projects so this organization can continue to increase the business value they are getting from open source?"

Improving security and resilience over time

With Tidelift in place, the organization had the visibility into their open source dependencies to further focus their development team's time and efforts. Tidelift could lead the effort of monitoring and ensuring accountability for the independently maintained open source packages in their application. With the increased predictability around their open source software supply chain, the customer had more time and resources to focus on their internal developers' development lifecycles. The customer now knew:

- Which open source packages—and their dependencies—are good to adopt any time a developer is considering bringing a new package into the application
- How to ground a strategic conversation about technical or maintenance debt in concrete data with reports showing packages that are end of life, abandoned, or otherwise unmaintained
- How to prioritize a set of recommended actions that remove risk of violating their security and licensing policies

Next, this customer needed to ensure they'd continue to improve the predictability and reliability of their application over time. Here's how Tidelift has partnered with them in this effort:

In 2022, when this customer first came to Tidelift, 22% of the application components were already backed by corporations or foundations (and that percentage has remained fairly stable at 23% today). Tidelift has historically focused on providing income primarily to independent open source maintainers rather than those backed by corporations or foundations because (as Tidelift co-founder Luis Villa describes in his blog post Paying maintainers: the HOWTO), smaller, independently maintained projects often have fewer avenues for funding, are sometimes less objectively well-known, so the money can have a larger impact on improving security and reliability.

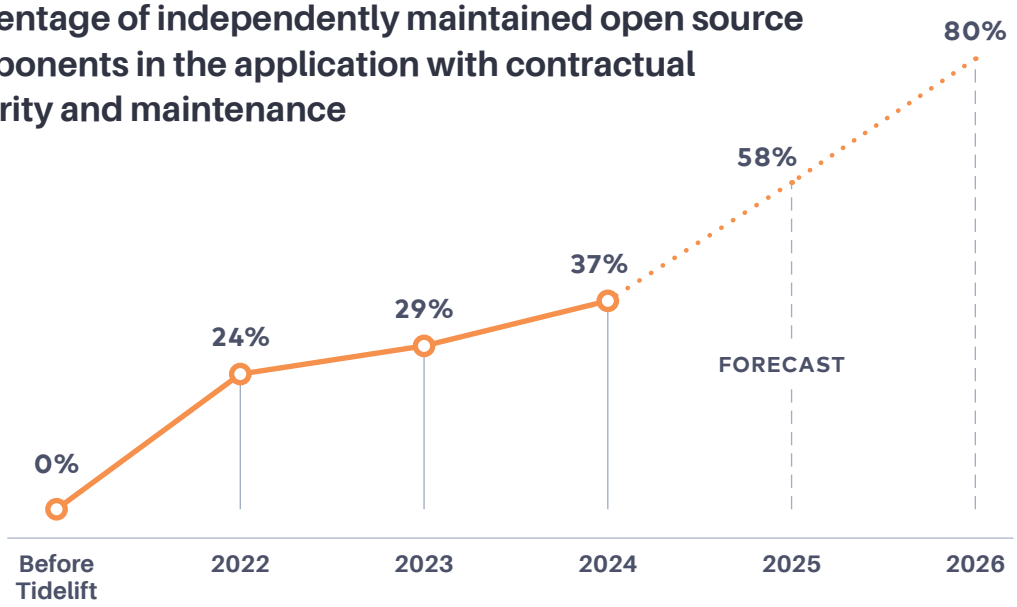
Conversely, projects backed by corporations or foundations may have some of the maintainers on the payroll, have more avenues for funding, be more well-known, and thus may be less "at risk" than independently maintained projects. One important caveat: this does not mean organizations should assume that the foundations or corporations backing these components have any contractual obligation to ensure they meet secure development requirements, but they can often be considered less "at risk" than components backed by independent open source maintainers who have less support and attention for their work.

Because Tidelift already had contractual commitments with the maintainers of 19% of the application's dependencies on day one, the recruitment effort focused on the remaining application components whose independent open source maintainers were not yet partnered with Tidelift.

The chart below focuses only on the packages from independent open source maintainers (meaning, it removes the packages that are already backed by foundations or corporations). From a starting point of 24% of independently maintained components in the application with contractual security and maintenance through Tidelift, the percentage continued to rise as Tidelift recruited more maintainers because of package usage by this customer and others.

By 2024, coverage of this application by Tidelift-partnered maintainers had risen to 37%, and is predicted to continue to rise to about 80% by 2026 as more independent maintainers are recruited. This represents almost "full coverage" because not every maintainer is interested in getting paid to implement secure software development practices, and if the customer would like to see this percentage rise beyond 80%, it might require rearchitecting some of the components out of the application whose maintainers are not interested in income for their work. This projection is based on historic growth and income acceptance rates, as well as ratio of maintainers to packages maintained in this particular ecosystem.

Percentage of independently maintained open source components in the application with contractual security and maintenance



Business impact

We led up front with this headline, but it bears repeating. Because this organization was able to reduce its reliance on abandoned, end-of-life, or otherwise insecure open source packages it was able to:

- Save $1.1 million of organizational time across engineering, legal, and security that would have been spent on requirements research and engineering implementation time
- Reduced application risk by turning 37% of this customer's independently maintained packages from an "unknown future" to reliably secured and maintained, with that percentage continuing to grow

Let's dive a bit deeper into the business impact beneath the headlines.

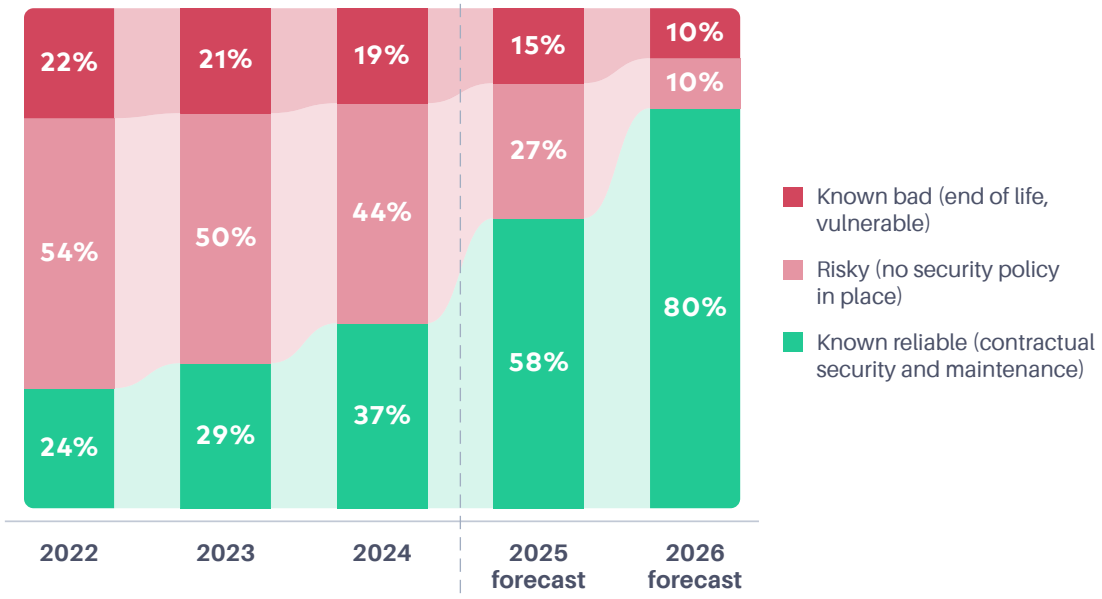
Reduced costs

With an abundance of open source packages available to modern software developers, it may seem like it would be easy to simply remove an open source package that is, or appears to be, abandoned. But it is rarely that easy. The package may be deeply embedded into the application and its dependency graphs, so the rework required may be extensive. Also, there are other costs to consider, including the time to research and test replacements, document what has changed in the application, training the team on the replacement, all while ensuring customer experience continuity.

> "We're just starting to account for the cost of ripping out a package as budgets tighten, and we know it's **large**. A proof of concept, legal, security, and other checks are required. Does the new package come with support? Is the package stable? What does the CVE history and licensing look like? What is the adoption rate of the library—are we the first ones to adopt? If any of these answers are unsatisfactory, it can result in restarting the research and approval processes. A minimal change can drag out 3 months, average 6 months, and replacing a platform or major framework can take a year.
>
> Then we have to plan a transition. How we will migrate a feature to a new package or framework. Because we have to ensure stability and a good experience for our end customers. It depends on the amount of change. If it's a smaller change, we'll roll it out within a quarter. If it's more, it takes half a year. If it's urgent, due to a vulnerability that is presenting risk, there are multiple tradeoffs that must be made." — Senior IT Director

When an abandoned package turns up with an unfixed urgent vulnerability, either directly or through a transitive dependency, it can also cause a fire drill. For a deeply nested transitive vulnerability, this becomes a multi-week or even multi-month cost where the business is still carrying the risk until that risk is eliminated. For two high profile examples of software supply chain security threats from a single package that created administrative overhead for many companies, read more about colors and left-pad.

Independently maintained packages with reliable income provide stability, code maintenance that reduces security, engineering, and legal costs.



Developer time spent patching known vulnerabilities is reduced (red) as Tidelift guides developers to safer packages, including their dependency graphs. At the same time, Tidelift reduces the uncertainty of broader software supply chain security threats (pink) by partnering with independent maintainers and paying them to implement secure development practices (green), and guiding developers to safer packages.

As this chart above shows, Tidelift has been able to reduce these potential costs to the business significantly already in this one application. With plans to continue improving contractual security and maintenance coverage over the next two years, the organization can expect to significantly reduce costs related to unplanned work and emergency fire drills that suck time away from other important business priorities. This organization has already saved an estimated $1.1M in costs so far, with the savings expected to rise as maintainer coverage continues to grow.

Reduced risk

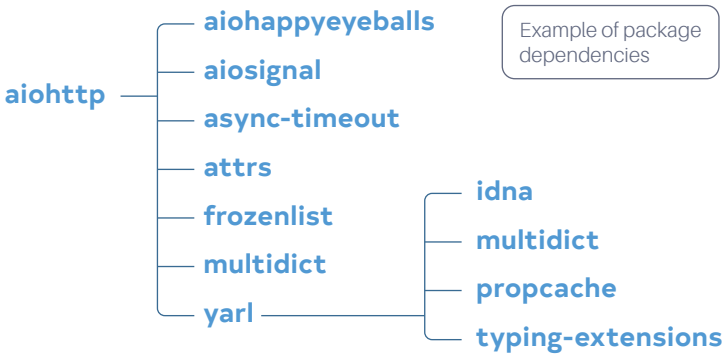
The number of reported CVEs is rapidly increasing each year, and the rise of AI capabilities to detect and report vulnerabilities in open source projects will make this number grow even faster. Maintainers are telling us that many of these AI-produced reports are time-wasting false positives today. Problems at this scale require innovative solutions—security and development teams' capacity simply can't keep up.

> "The increase of spam PRs, comments, and false positives from AI tools and users has been enormous and very frustrating." — Maintainer, via Tidelift 2024 maintainer survey

Tidelift's partnered maintainers are the front lines of managing their dependencies on behalf of this customer's internal development team. Maintainers are able to mark false positives and apply patches—or move off of a dependency when a patch is not available. This is driving down risk on behalf of the customer.

What happens when CVEs enter an application transitively? Software supply chain security is a true *chain*, with many inaccessible nodes. Customers may decide to stop using a package directly, as this customer did multiple times, only to have it show back up transitively, through another package they depend upon. Even if teams are putting measures in place to control and approve the surface area of open source they must manage, they have no control over how many dependencies those packages bring in, or if those new packages meet their quality requirements. This customer saw Tidelift as the best way to reduce this risk.

Independently maintained packages with reliable income eliminate software supply chain risk that the customer's engineering team cannot otherwise reach.



The surface area of potential risk is much larger than a flat list of all direct dependencies reveals. As with this example, the maintainer of a direct dependency (like aiohttp) is responsible for ensuring the entire dependency graph under them is safe from risk. This is why ensuring maintainers are being paid to monitor their dependencies' security is critical to reducing application risk.

In this example above, aiohttp's graph is pulling in 16 dependencies where the customer cannot influence dependency health on their own and has limited visibility. But because the maintainer of aiohttp is being paid by Tidelift to ensure their dependency graph is kept free of risk, the risk to the customer's application from transitive dependencies is reduced.

Next steps

This customer was able to establish a real partnership with the upstream developers who maintain the majority of their code. Tidelift's partnered maintainers reduced the customer cost of patching endless vulnerabilities and ripping out and replacing packages. The result was hundreds of potential hours of assessment, research, approvals, and rework saved. The customer reduced unplanned vulnerability fire drills that abandoned or unmaintained dependencies create. These partnered maintainers also ensured that transitive package risk was removed for the customer.

Based on Tidelift customers' reporting an average of 4.5 months for cross-functional teams to review a major package change, we estimate the value of this time saved to be *a minimum* of $1.1m. The scale of this cost reduction for an organization cannot be overstated. And, this same set of Python packages is used across **3,166 other Python applications spanning multiple application teams** for this customer.

With this strategy in-hand, the customer maximizes the ROI of their whole organization's time spent on open source lifecycle maintenance. This ensures that the customer stays competitive in the market, with their developers building functionality that drives revenue instead of wasting time on open source maintenance tasks. From adopting new packages, to monitoring and moving away from problems, Tidelift is their trusted partner helping reduce risk from bad open source packages and ensuring that the open source used in their applications keeps getting better.

Tidelift is also continuing to run analysis for packages in their applications where there are not yet maintainer partnerships in place, and the next recruitment class has been identified—*including* one of the customer's currently abandoned packages!

This customer will continue to be able to accurately forecast pricing, because they can stay focused on investing their time and resources in improving their internal code, while Tidelift and its maintainer partners take the lead on ensuring their open source code and software supply chain continues to be secure and resilient, now and into the future. ■

PART 2

How paying maintainers improves project security and resilience

The case story we just shared showed how one organization is able to reduce organizational risk and improve operational efficiency—when they invest directly in their open source supply chain by paying maintainers.

But this is not an isolated datapoint. A growing body of evidence shows that paying maintainers to ensure their projects follow secure software development practices is an effective way to reduce security risk while keeping packages resilient over time. In this section, we'll share some of the most compelling data showing that paying maintainers is an effective strategy for reducing open source-related risk in their applications.

Paid maintainers are 55% more likely to implement critical security and maintenance practices than unpaid maintainers

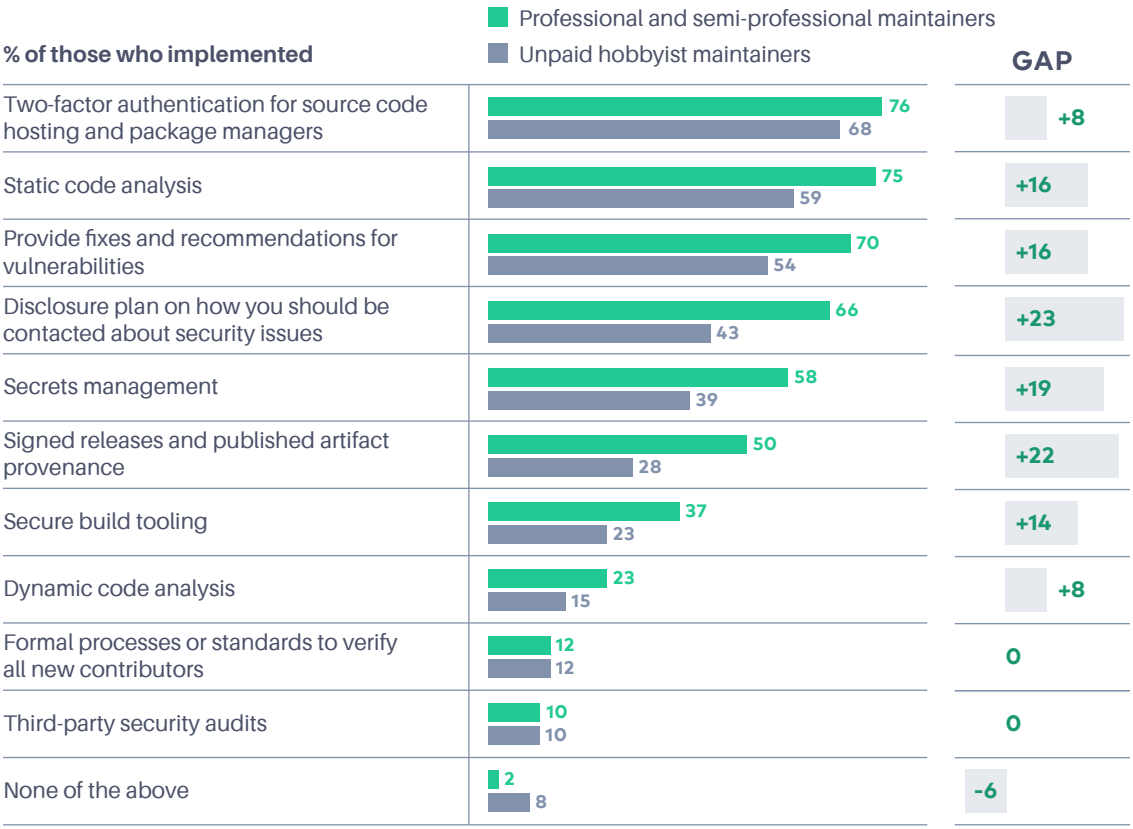
Each year, Tidelift fields a survey of open source maintainers and reports the results as part of the annual state of the open source maintainer report. For the most recent survey in 2024, over 400 maintainers participated and shared details about their work, including how they fund it, who pays for it, and what kinds of security, maintenance, and documentation practices they have in place today or would consider in the future.

For the purposes of studying the impact of paying maintainers, we asked maintainers about the security and maintenance practices they follow, including a comprehensive list of important practices like those found in the OpenSSF Scorecard, the NIST Secure Software Development Framework, and even those required of Tidelift-partnered maintainers. We then sorted the answers they gave us by whether the maintainers were paid professional or semi-professional maintainers or unpaid hobbyists.

Nearly across the board, paid maintainers are 8–26 percentage points (or, on average 55%) more likely to implement many critical security and maintenance practices than unpaid maintainers.

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

The top security practices implemented by paid maintainers include two-factor authentication (76% compared to 68% for unpaid maintainers), static code analysis (75% vs. 59%), providing fixes and recommendations for vulnerabilities (70% vs. 54%), security disclosure plan (66% vs. 43%), secrets management (58% vs. 39%), and signed release and published artifact provenance (50% vs. 28%).

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)

% of those who implemented	Professional and semi-professional maintainers	Unpaid hobbyist maintainers	GAP
Formal policy about backwards compatibility	59	39	+20
Reproducible and verifiable build processes	58	50	+8
Code peer review process with multiple reviewers	53	27	+26
Defined dependency management process	50	33	+17
Formal processes or standards to prioritize the order in which pull requests and issues are addressed	17	13	+4
Formal processes or standards to verify all new contributors	12	12	0
None of the above	15	26	-11

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

The top maintenance practices implemented by paid maintainers include a formal policy around backwards compatibility (59% compared to 39% for unpaid maintainers), reproducible and verifiable build process (58% vs. 50%), code peer review process with multiple reviewers (53% vs. 27%), and a defined dependency management process (50% vs. 33%).

MAINTAINER IMPACT STORY: APACHE COMMONS

Maintainer Gary Gregory used income from Tidelift to triple his output, including being able to respond more quickly with bug fixes and enhancements. The picture here shows clearly how his activity on Apache Commons Lang increased drastically when he started getting paid for his work.

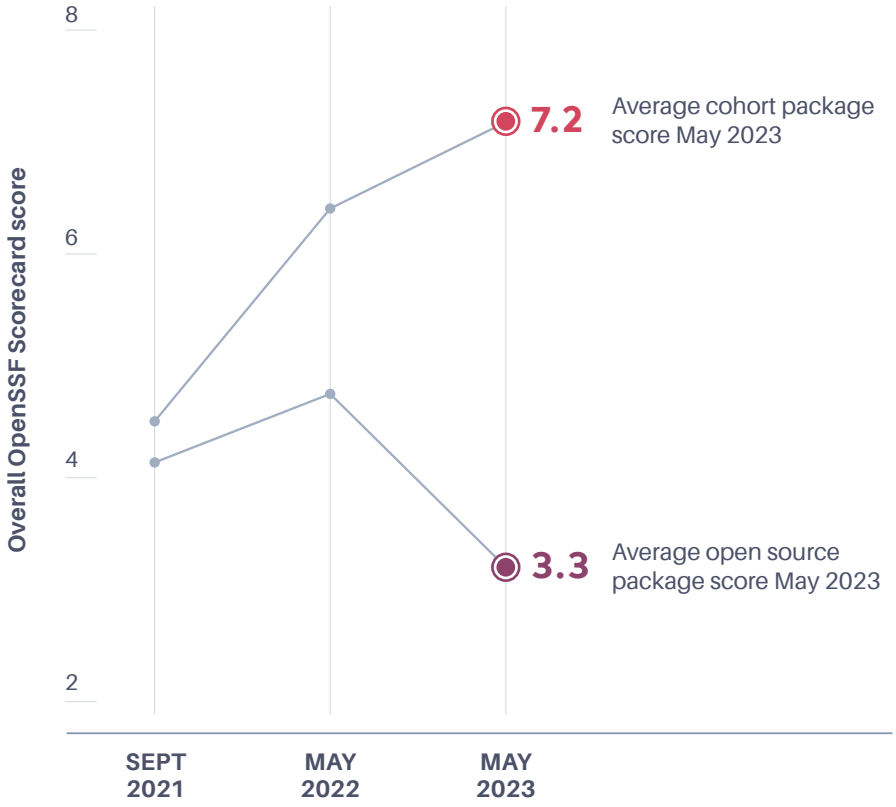
"...my involvement with Tidelift has caused me to create a tighter feedback loop for any of these issues," Gary said. "I've created a lot more releases, which means that all the bug fixes and enhancements go out a lot quicker and with a lot more frequency. Whereas in the past, I would just release versions of components whenever I felt like it, or whenever I needed them."

Paying maintainers to improve OpenSSF Scorecard scores

In last year's maintainer impact report, we reported the results of a project where we paid a cohort of maintainers specifically to complete a set of defined tasks that would improve the project's OpenSSF Scorecard scores. The results clearly showed that Scorecard scores will improve dramatically when maintainers are paid to implement security practices. We found that the average score at the end of the project was 7.2 (out of a possible 10) from a starting point of just over 4. Meanwhile the average open source package not a part of the project was 3.3.

Tidelift's model ensured that this cohort already had baseline security practices in place when the pilot started (May 2022), and used a combination of financial incentives and hands-on help to drive adoption of additional security measures (May 2023).

OpenSSF scorecards over time: focused cohort vs. all assessed open source packages



Other research showing the security benefits of paying maintainers

Tidelift is not the only organization studying the impact that paying maintainers can have on the health and resilience of open source projects. Recently, in their 10th annual State of the Software Supply Chain report, Sonatype shared some of their findings on the subject as well (emphasis ours).

From the report:

> "Paid maintainers show a clear lead in security practices. **Projects with paid support are nearly three times more likely to have a comprehensive security policy formed through best practices like those verified through the OpenSSF Scorecard project**, suggesting better vulnerability identification processes. At the same time, **non-paid packages tend to accumulate more vulnerabilities, with paid packages having only a third of the unfixed vulnerabilities seen in non-paid ones**. Additionally, **components with paid support resolve outstanding vulnerabilities up to 45% faster and have half the vulnerabilities overall**. This data highlights that incentivized maintainers produce more secure and efficient outcomes.

Meanwhile, the Digital Forensics Lab at the Atlantic Council has also studied the issue and reported their findings in the article O$$ Security: Does More Money for Open Source Software Mean Better Security?

Here are a few quotes from that report summarizing their findings related specifically to whether projects with funding were able to improve their security posture as measured by OpenSSF Scorecard scores (again, emphasis ours):

> "In short, **these findings indicate with moderate confidence that there is a meaningful connection between more open-source project funding and improved security posture. Some practices are strongly associated with funding, and more funding generally correlates with more dramatically differentiated security practices**.
>
> If these results indicate that funding leads to better security practices, the causal explanation is relatively simple and intuitive. **More money for maintainers and even developers means more flexibility in dedicating time to project management, which can include developing security policies, remediating vulnerabilities, using better permission tokens, and including CI testing, fuzzing, signatures, packaging, update tools, and more—all of which increase Scorecard scores. Additionally, funding may help purchase either tooling or project services that would similarly contribute to security posture**."

MAINTAINER IMPACT STORY: URLLIB3

urllib3 is a HTTP client for Python that is used—directly or indirectly—by nearly 1.5M direct dependent repositories, and is regularly among the top 3 downloaded Python projects. Seth Michael Larson, Andrey Petrov, and Quentin Pradet—the team of maintainers behind urllib3—used their income from Tidelift to uplevel their security practices, including hiring an additional maintainer, Illia Volochii, to work on important projects, like securing maintainer access with two-factor authentication, automating release processes for consistency and security, and achieving reproducible builds to prevent supply chain attacks.

Through these efforts, urllib3 became the first Python project to achieve an almost perfect OpenSSF Scorecard score of 9.6 out of 10.

The bad news: 60% of maintainers are still not paid for their work

So far, we've shared mostly good news. Maintainers who are being paid for their work implement more security practices that your organization can benefit from. So what's the catch?

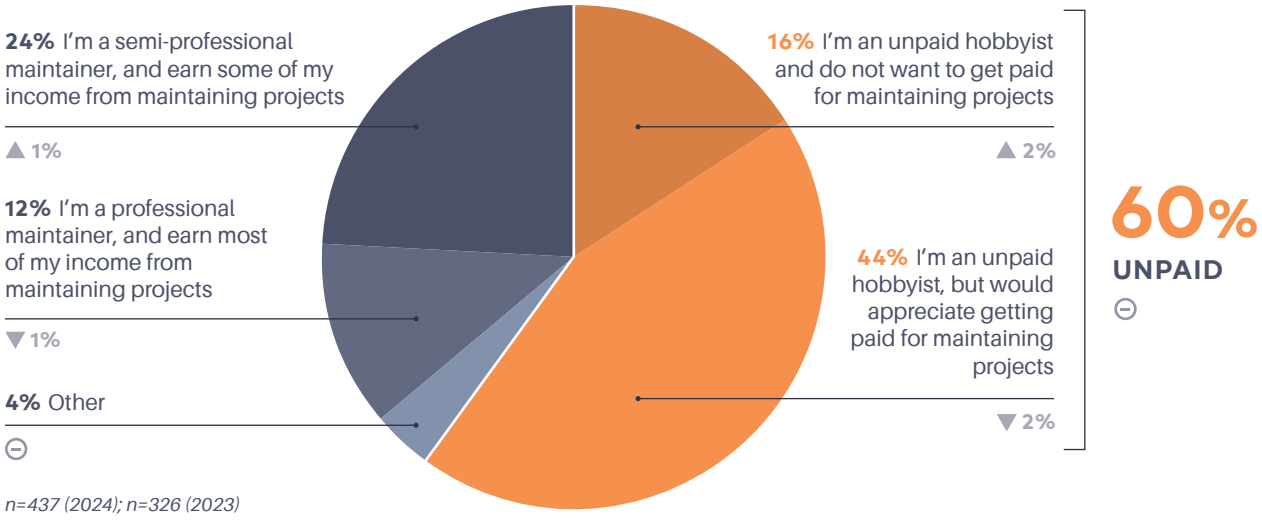
Organizations are under-investing in the security of their software supply chain.

The chart below shows that 60% of maintainers categorize themselves as unpaid hobbyists. The percentage of maintainers saying they earn most or all of their income from maintaining projects is only 12%, and the percentage of semi-professional maintainers is only 24%.

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)

For many people, especially those whose organizations rely heavily on open source, this finding is shocking. After all, it is not safe to assume that unpaid hobbyists have the time or motivation to ensure their projects follow the secure software development practices you might expect from your own organization's code, and that governing bodies are starting to require.

So beyond the risk this can cause to the security of your organization's applications, what are the other ramifications of the majority of maintainers being unpaid?

Many maintainers have quit or considered quitting, which threatens the security of your software supply chain

Pressure on open source maintainers to do more continues to rise each year, as expectations from industry and government rise. As part of our survey, we asked maintainers to tell us what they dislike most about their work and the results were clear: maintainers feel like they are not financially compensated enough for their work, they feel underappreciated and like the work is thankless, and they feel like being a maintainer adds to their personal stress. They are asked to comply with requirements they don't have time for, and their users are too demanding and expect too much of them.

As one maintainer said:

> "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."

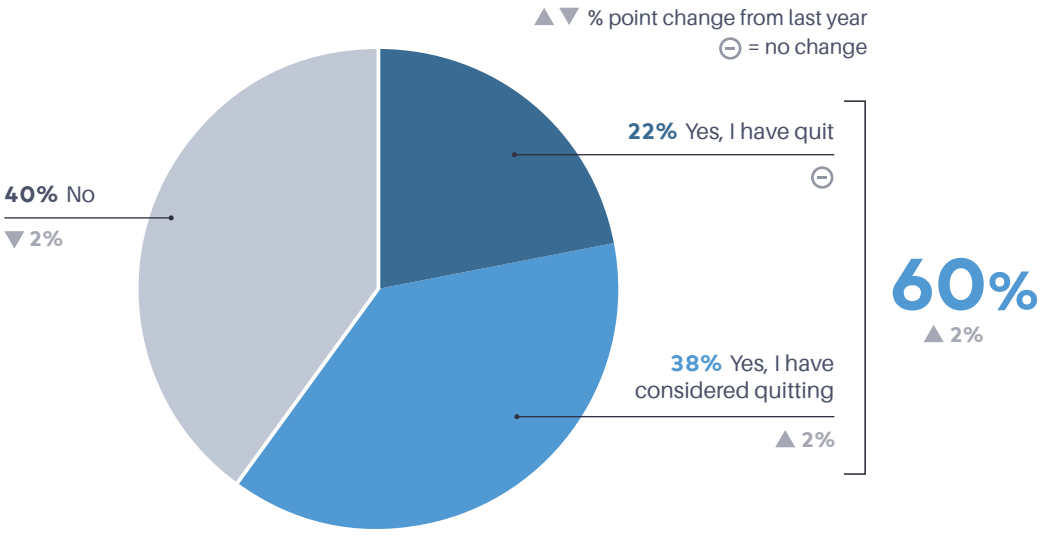
Another maintainer complained about the sense of entitlement many users have.

> "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."

So what does this dissatisfaction with the current state of their maintainer work mean for the projects your organization relies on? It means that many maintainers may be abandoning their projects soon... if they have not already.

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)

Each year we've asked maintainers if they have quit or considered quitting their work, and the results are startlingly similar. 60% of maintainers have quit or considered quitting maintaining a project.

This means that throughout your open source software supply chain, you may have packages that are at risk of becoming insecure or abandoned, simply because their maintainers no longer have the motivation to continue. **To put this into stark relief, in a 2023 snapshot of application libraries reviewed by Tidelift, 26% of all end-of-life releases in use carried at least one vulnerability.**

As one maintainer put it:

> "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 declining maintainer motivation to continue working on their projects creates a dangerous—and often unseen—security risk for organizations using open source to navigate. And it is another powerful reason why paying maintainers can have an enormous impact on improving the security and resilience of your organization's software supply chain. ■

MAINTAINER IMPACT STORY: MINIMIST

A while back, npm made the decision to start requiring maintainers of JavaScript packages to use two-factor authentication. Some JavaScript maintainers weren't so enthusiastic—requiring 2FA was one additional burden on them that many of them weren't being compensated to take on.

For one particular maintainer of several high-profile packages, including one called minimist, who went by the name "substack," this new requirement was a problem. As a result, he sought out to delete his entire GitHub account.

One of Tidelift's maintainer partners, Jordan Harband, spoke with substack and offered to take over maintenance on all of the packages. But when it became clear that there was not an easy "one click" way to transfer ownership of the rest of the packages, substack lost interest and deleted his account.

Jordan recalled watching as the packages started to disappear, one by one, until they were all gone. He reached out to GitHub to see if there was any way to undelete the packages and transfer ownership of them, and called in Tidelift to help make the case for him to take over ownership of the packages.

Eventually, with the approval from substack, Jordan was able to get complete ownership of over a dozen of the impacted packages.

Even better, because he was paid to take on this work, Jordan was able to bring in an experienced co-maintainer for minimist, who also shares in the revenue and provides extra support to ensure that it is no longer a solo maintainer project like it was before, making the future resilience for minimist even greater.

How Tidelift can help

The core thesis of this impact report is that organizations can proactively improve the security and resilience of their software supply chain by paying maintainers of the open source projects they rely on to follow secure software development practices. If you are convinced, and want to understand what next steps your organization can take, Tidelift is here to help.

If you are generally interested in learning more about some of the key principles for how to pay maintainers effectively, a good place to start is Tidelift co-founder Luis Villa's blog post Paying maintainers: the HOWTO that we mentioned in part one. In this post, Luis covers which kinds of projects to pay and what you should pay the maintainers of these projects to do.

But if you want a partner to help you systematically reduce open source-related risk the way that we helped the customer in part one, you'll want to work directly with Tidelift.

Tidelift helps organizations increase the security and resilience of their applications by partnering directly with open source maintainers. Tidelift is the only company that pays them to:

- Implement industry-leading secure software development practices and validate the practices they follow so organizations can have the same confidence in the security of their open source that they have in their own code
- Contractually commit to continue these practices into the future so that organizations can confidently make long term investments in the packages they use

Tidelift helps organizations use open source for building their applications with confidence. Our customers:

- Reduce security risk by eliminating attack entry points through abandoned, end-of-life, or insecure open source packages
- Improve productivity by reducing vulnerability fire drills from insecure or undermaintained packages
- Improve application quality by ensuring they build with healthy and resilient open source packages
- Increase operational efficiency by saving costly manual package evaluation and remediation time

Software security concerns continue to be a growing threat to organizations' revenue, data, and customers. Meanwhile, global cybersecurity regulations continue to expand, with most companies preparing to be able to report on the software ingredients used in their products, their secure software development practices, and their response to cyber incidents.

This means that we are entering a new age of accountability, where businesses will need to invest time and money to comply with new external requirements while also increasing the internal diligence they put into hardening their software supply chain to protect their own assets as well.

If you are interested in having a strategic partner in this effort, one that can not only help you get better information about the current security profile of your open source software raw materials, but also help you ensure they continue to become more secure and resilient, please reach out to us. We'd love to help. ■

TIDELIFT.COM