Security of Open Source Software

A Survey of Technical Stakeholders' Perceptions and Actions

FEBRUARY 2023

Centre for Internet & Society

AUTHOR:
Divyansha Sehgal

REVIEWERS:
Divyank Katira & Isha Suri

CONCEPTUALISATION:
Puthiya Purayil Sneha

SURVEY ORGANISATION:
Operations Research Group, India

COPY EDITING:
The Clean Copy

LAYOUT DESIGN:
Indumathi Manohar

This work is funded by the 2020 Digital Infrastructure Fund, by the Ford Foundation, Alfred P. Sloan Foundation, Open Society Foundations, Omidyar Network and Mozilla Foundation in collaboration with the Open Collective Foundation.

Contents

Executive Summary

Open-source software (OSS) components are largely assumed to be secure due to their open nature.

However, that is not always the case. Of late, there has been an increased incidence of software supply-chain issues, with some industry reports estimating a 300% increase in attacks that exploit existing vulnerabilities between 2020 and 2021.

This report by Centre for Internet and Society surveys technical stakeholders to determine how they select OSS components to use in their projects and how they think broadly about the security of the projects they create.

Highlights at a Glance

90% of respondents **work in companies with a dedicated team** responsible for the security of software. **80% of them do not carry out any further security checks** on an OSS once it has been approved for use by their security teams.

80% of respondents **see comprehensive documentation as an important factor** when selecting an OSS for use.

70% of respondents **report validating dependencies** in their selected open-source software component.

50% of respondents **consider how actively an open-source software is maintained** before selecting it for their projects.

40% of respondents **do not anticipate accidental exploitation of vulnerabilities** or expect malice from bad actors when they create software.

30% of respondents **report not doing any post-release maintenance** on the OSS component used and deployed.

Takeaway 1:

Most stakeholders depend on security teams to thoroughly evaluate the security of the software they have selected for use. Once an OSS project has been approved for use by the in-house team, no further security checks are deemed necessary.

Takeaway 2:

80% of technical stakeholders look for project documentation, and 70% seek an active community around an OSS component. Only around 50% look for evidence to suggest that the project is actively maintained.

However, this search for documentation, community, and regular maintenance is motivated by a consumer outlook towards OSS that prioritises the ability to solve problems and treats the community as a 'help desk' rather than a group of people working together to create a useful service.

Takeaway 3:

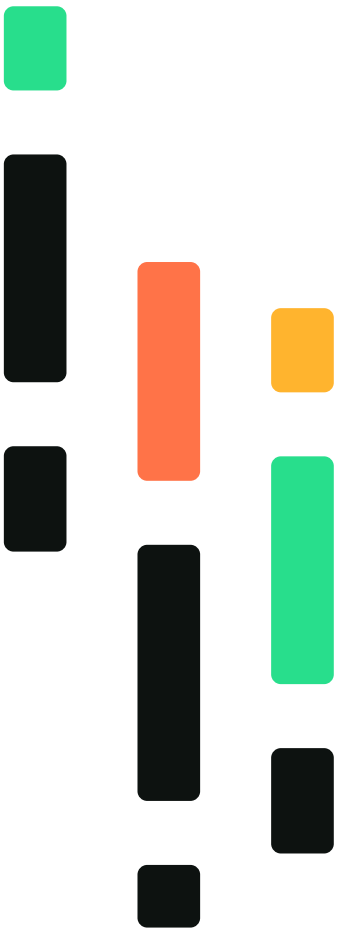
Over 40% of people do not consider accidental exploitation of vulnerabilities or intentional malice by bad actors as a threat when they create software.

Takeaway 4:

Post-release maintenance of tools is quite common, with 70% of technical stakeholders keeping track of and updating the OSS components they've used in some way.

Introduction

Open-source software are the building blocks of all modern consumer and business facing software products. Most software applications are created by re-using and modifying existing components for specific use cases.



Open-source software (OSS) components are largely assumed to be secure due to their open nature. Since code is free to be accessed and modified, it is assumed that there are a large number of eyes it. Thus, an issue in the code would be caught by someone in the ecosystem. However high-profile bugs, such as Heartbleed which stems from a widely used open-source web-encryption library, show that this is not always the case.[1] Bugs and software vulnerabilities in open-source libraries can propagate downstream to all software projects that use the component.

Open-source software are the building blocks of all modern consumer and business facing software products. Most software applications are created by re-using and modifying existing components for specific use cases. The collection of OSS components used to create a software application comprises that project's supply chain.

Of late, there has been an increased incidence of software supply-chain issues, with some industry reports estimating a 300% increase in attacks that exploit existing vulnerabilities between 2020 and 2021.[2]

The SolarWinds attack that affected thousands of organisations, including the US government, showed yet again how vulnerable the software supply chain actually is.[3]

The reasons for this have also been discussed in detail in both academia and industry. Nadia Eghbal's comprehensive report outlines the following issues[4]:

- a vast majority of software projects are maintained by individuals who do not have the resources to keep them secure;
- most people who use OSS do not contribute to the libraries they use;
- there are more consumers than there are maintainers of OSS;
- developers are not incentivised to contribute to the community since neither tech companies nor governments invest enough resources into the ecosystem to sustain this open digital exchange.

Yet, OSS projects are thought to be an integral part of enterprise software development, with many professionals pointing to its assumed secure nature as a reason to increase adoption.[5]

Our study aims to further explore this apparent contradiction between users' perceptions and the reality of OSS security. Specifically, we look at how professionals in the IT industry select the OSS components to use in their projects and how they think about the security of the projects they create.

01

Methodology

For this study, we interviewed 410 IT professionals across industries, including e-commerce, finance, automotive, energy, health, hospitality, and insurance, to understand their perceptions of OSS.

The survey was conducted in partnership with Operations Research Group, India, an India-based survey agency, to administer the survey virtually at scale.

We followed a purposive sampling method with respondents drawn from a variety of technical implementations roles: developers, product managers (PMs), technical PMs, technical team leads, etc. The interviewees were selected if they have been working for an IT company for at least six months and have experience selecting and using open-source projects or components for their work. To participate in the survey, the respondents received an honorarium in exchange for their time.

A survey tool was created to cover the socio-technical aspects of software security including the security culture of the organisations that employees belong to and the features of the code they evaluated (explained in detail in the following section). The interviewers conducted a semi-structured, in-depth online interview with participants over Zoom and probed their choices further, which supplemented quantitative answers with personal context. Prior to the commencement of interviews at scale, a pilot survey was conducted with **30 participants** that helped identify issues concerning survey length and complexity.

We used the feedback to further refine the survey.

Development Of Survey Tool

To create the survey tool, we considered the literature on how stakeholders select OSS components they use and how they determine trustworthiness and fit.

Prior research shows that practitioners consider a variety of factors including functionality and fit for the task, cost, license types, community characteristics, documentation, and security.[6]

Reputation, performance, fit, and community metrics are repeatedly mentioned as factors that determine the choice of an OSS.[7,8,9,10,11] The security of OSS also finds repeated mention, but with much lesser frequency than other factors across the literature reviewed.

Looking specifically at the security considerations of OSS, the literature shows that most research focuses on the technical aspects of software security.[12] While important, these aspects do not offer a complete picture since they ignore the context of a developer's environment and organisational preferences.

Software Assurance Maturity Model (SAMM) by the Open Web Application Security Project (OWASP)13 is a model that considers the socio-technical aspects of software security and sets guidelines for organisational priorities depending on their scale. By combining the OWASP SAMM with other best practices and software security frameworks, we created a survey tool that seeks to assess the stakeholder approach to OSS security across the following factors:

1. Organizational Security Culture

The security culture of an organisation is defined as the set of values and cultural attitudes towards security embodied by employees and encouraged by the management and executive decision makers.

Security culture depends on various factors including the size and maturity of the organisation, the type of product they create, the business priorities of the organisation, and personal security preferences of employees. It determines how employees think of the security of the work they use or create, how they prioritise it in their daily routines, and the extent to which they seek out and follow best practices.

2. Evaluation Checks

A combination of academic and grey literature shows that the metadata features of code and its context influence stakeholders' trust in an OSS component and may often stand in as a proxy for the estimated security of this component.[14,15,16] We asked stakeholders if and how they consider the following contextual information about the code while they're selecting an OSS component to use:

Project Documentation: Project documentation is any and all the text around a project that is written by the OSS maintainers that is useful for the reader to understand the code. It may include user guides and API documents that explain the functionality expected, code samples and structure, the licenses needed, and any other information that the OSS developer deems necessary for the user to know. There are no enforceable rules around code documentation, and they can often run the gamut from scant to comprehensive.

Project Reputation & Popularity: The reputation and popularity of an OSS component play an important role in evaluators' decisions on whether to adopt it. While this is a less straightforward measure than the presence of documentation, it can be determined via metadata features that provide metrics like the number of downloads, frequency of code commits and issue resolution, code quality, recency of commit, etc.

Community: The people that use and contribute to an OSS component may gather in online forums, issue trackers, messaging platforms, and other spaces to discuss problems, raise questions, solve issues that users run into, and generally help each other get started and troubleshoot issues with the code. This sort of open discussion creates community knowledge that is helpful for new and old users alike to keep track of issues and address security vulnerabilities when they arise.

3. Adoption Checks

Once a component has been evaluated regarding its ability to provide the required functionality, and stakeholders verify if the required contextual information such as documentation, metadata is up to their standards, we sought to examine how closely evaluators look at the code itself to determine its trustworthiness. We probed along the following lines:

- What are the security best practices that respondents follow?
- Do they look for third-party badges and certifications?
- Do they test the code and its dependencies?

4. Post-Release Checks

Code maintenance is an important aspect of any software project. Our survey tool included questions to understand the extent of security-related activity once the development process is complete and the code is being used by the intended users.

These verticals provide a comprehensive view of how stakeholders weigh the security of the OSS components they use and the products they create. The findings of our survey are also presented using these verticals.

Demographic Information

Our final participant pool spanned a variety of technical and non-technical roles, company sizes, and industries across India. Over 90% of our respondents worked in technical roles that involved coding as part of their daily jobs (Figure. 1).

The most common technical roles were software engineers,technology consultants, and data analysts. Non-coding roles included quality and testing analysts, project and product managers, and various types of non-technical team leads.

Most of our final respondents worked in large companies (over 500 employees). Only about 30% of the sample comprised employees from medium and small companies (Figure 2).

Figure 1: Over ninety per cent of our respondents worked in technical roles that involved coding as part of their daily jobs.

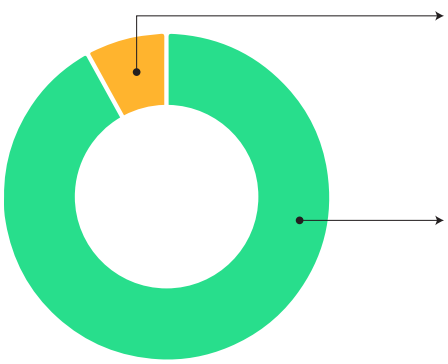
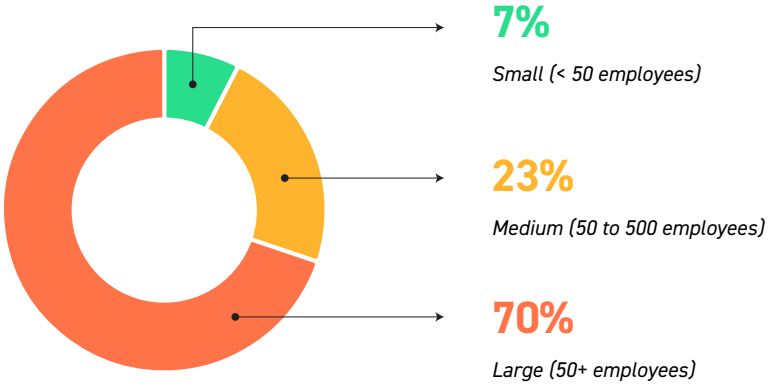


Figure 2: Most respondents worked in large companies, with 30% working in medium and small companies.



02

Findings and Discussion

Organisational Security Culture

Ninety per cent of the respondents in our sample said that their companies had separate teams that ensured the security of the software (Figure 4). We also observed a correlation between the presence of security teams and organisation size, with 93% of large organisations (500+ employees) having a formal security team while only 65% of small organisations (<50 employees) did so.

The security team performs a variety of functions, including setting best practices and automatically updating OSS software, as evident from Figure 5.

Figure 4: 90% of the respondents had a security team in their organisation.

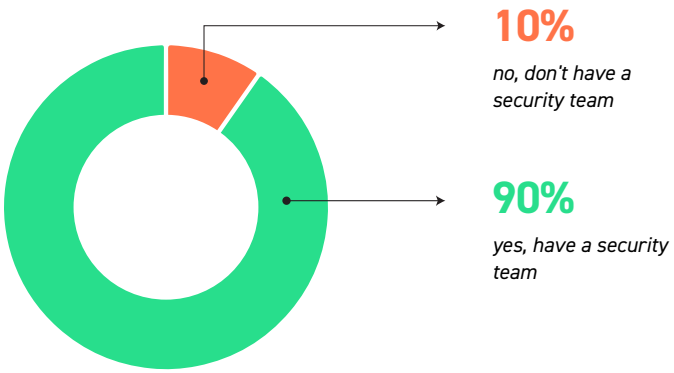
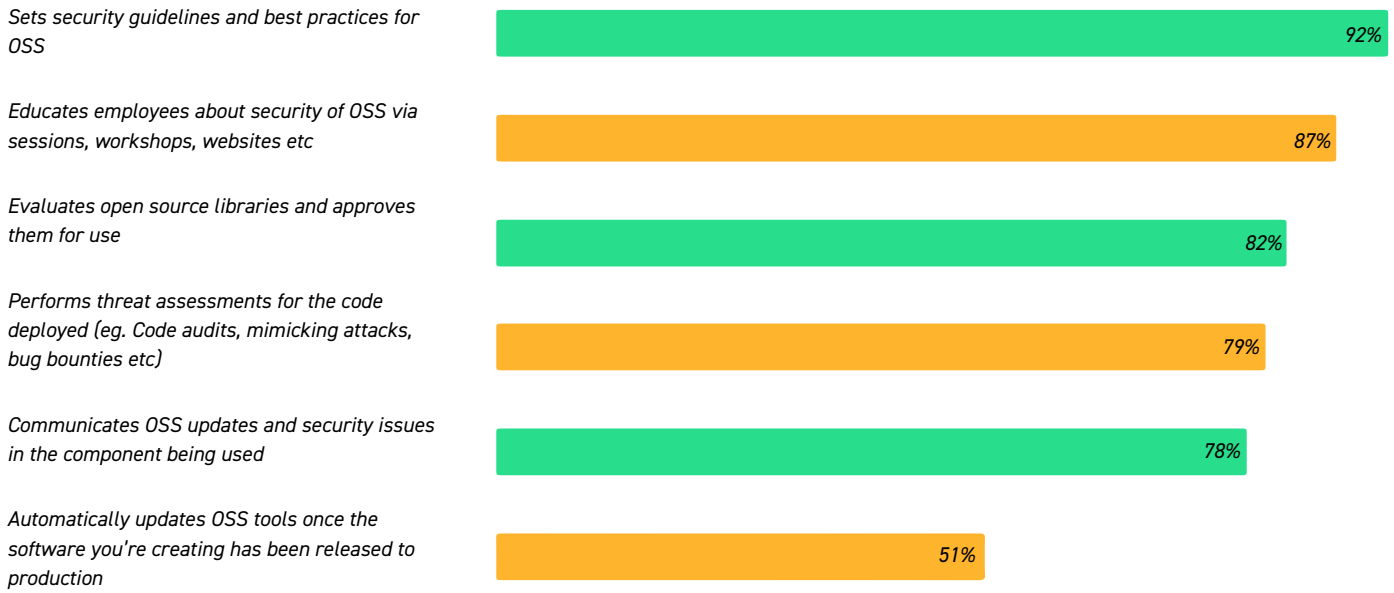


Figure 5: Functions performed by security teams.



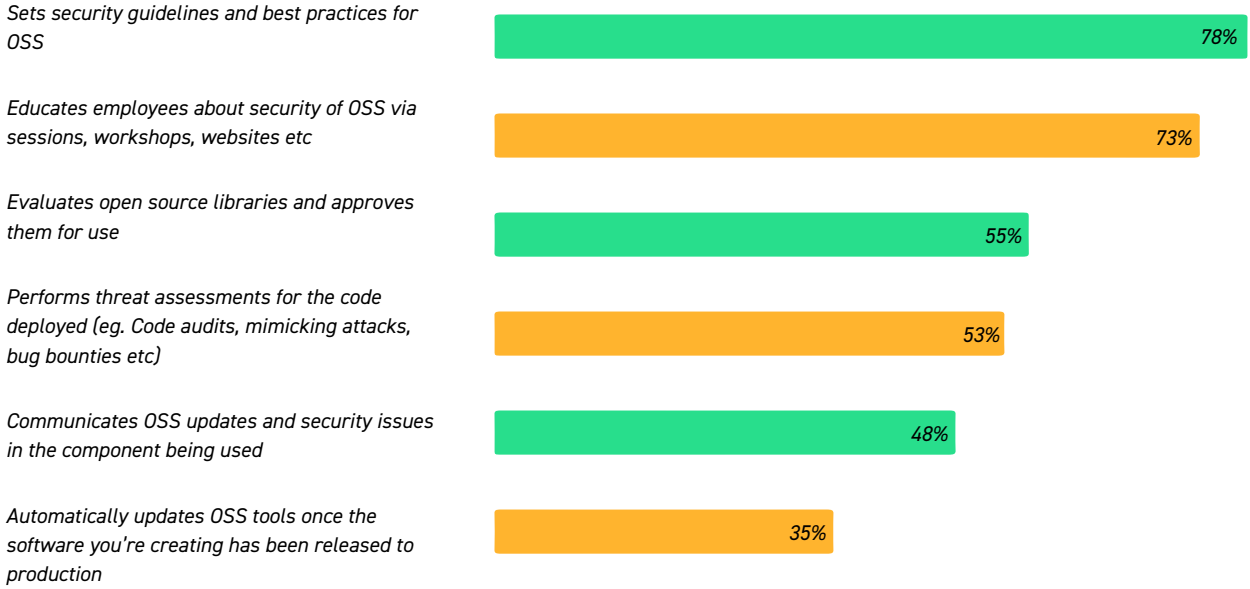
Most security teams set and communicate guidelines and best practices for their employees. Over 85% of employees also receive security education in various forms, including sessions, workshops, and links to relevant webpages. The participants in our survey cite both as extremely important functions performed by the team.

However, only about 50% of the security teams set automatic updates to the software components in use. It is the developer and their team's responsibility to stay on top of all information regarding the software used.

At the same time, the presence of a separate security team outsources the security responsibility to only them. Respondents reported performing checks for required functionality, documentation, and licenses before adoption. However, nearly 80% of our sample was confident in using an OSS component that has been approved by their company's security team, indicating that stakeholders do not do extensive security checks on code that has been approved by their security team. This direct mismatch between stakeholder and company expectations leaves room for software vulnerabilities to creep in and propagate.

When security teams are not present, these organisational functions do get performed, presumably by developers or the larger organisation, though to a lesser extent across all the functions measured (Figure 6).

Figure 6: Functions performed by the organisation without a security team



Automatic security updates were observed in only 35% of companies without security teams as compared to 50% in those with security teams. A steep drop was also observed in the security education role of the organisation, which fell from approximately 87% in companies with a dedicated security team to 53% in companies without one.

We also asked participants for other practices their security teams followed. Device surveillance and managing access to websites were the most frequently cited functions. Many respondents reported that their company devices logged keystrokes and recorded their action history. Security teams seem to be responsible for environment integrity and enforce this using a combination of surveillance and pre-emptive blocks on known 'unsafe' websites.

Only one participant volunteered that their security team also has emergency support functions in case of data breaches at their organisation. A few others noted that their security teams conducted personnel training that went beyond immediate software security and included tasks like imitating phishing emails to keep employees up to date on best practices that were not related to the software they were creating. This points to the fact that security teams not only authorise OSS components for use, but also perform a wide gamut of tasks pertaining to security education, device safety, and access controls to protect against unauthorised access to proprietary software. Software security also gives companies a justification for workplace surveillance practices.

Further research can look more deeply into the structure of security teams (including size, resources allocated, and physical and digital security roles performed) across organisations of different sizes and the effect it has on the overall security outlook of employees.

Takeaway:

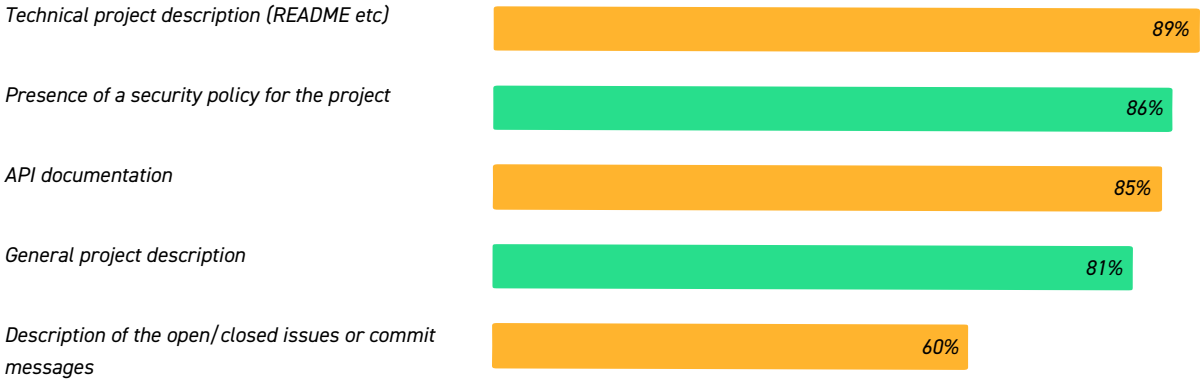
Most developers depend on security teams to thoroughly evaluate the security of the software they have selected for use. Once an OSS project has been approved for use by the in-house team, no further security checks are deemed necessary.

Evaluation Checks

Documentation

Over 80% of respondents said that they consider the completeness of project documentation when deciding what tool to use. As discussed, documentation comes in many forms (Figure 7) and is mostly used for functional purposes in the decision phase. Most stakeholders engage with technical descriptions and API docs. The presence of a security policy is also important, with 85% of the survey respondents reporting that they specifically look for the project's security policy.

Figure 7: Different forms of project documentation.



Community Factors

Community is an important factor as well. Employees are more comfortable using software if other people are actively using it. More specifically, stakeholders identify an active community around software components by assessing activity on forums like StackOverflow, GitHub, and Quora. When prompted, many participants pointed to the speed at which queries receive responses as an important feature in evaluating total community activity. To estimate responsiveness, stakeholders either pose questions in these forums themselves or check the number of open and closed questions to determine community activity. Employees also told us that they look for educational documentation online as the presence of teaching modules or advice documents from third parties is a good measure of community activity. Internal company knowledge about the OSS in question is also considered as it increases the possibility of getting advice and help from colleagues. A few users also mentioned checking commit history to see how quickly feature requests were delivered as usable features. This points to a consumer outlook in users of open-source software. Stakeholders seem less concerned with how the component gets created or maintained and more focused on functionality and access to a dedicated 'support team' for their problems.

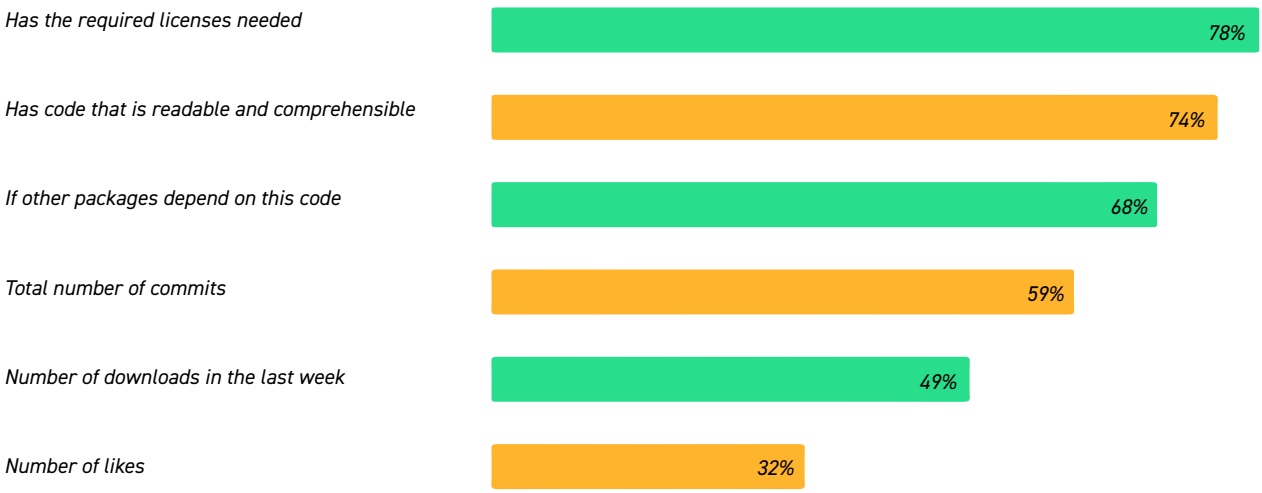
Popularity and Maintenance Metrics

Various metadata and contextual features can point to how popular and well-regarded an OSS component is in the community (Figure 8).

In our sample, the presence of correct licences inspires trust in 78% of people, followed by the readability of code at 73%. Knowing that other packages depend on this code also increases trust in the software (67%).

This is consistent with the reports we have seen in the community section of the survey as well: if colleagues with prior experience have advised the use of a particular OSS tool, evaluators are likely to trust it as well. Code owner/maintainer responsiveness is a somewhat less important factor in evaluating reputation, with only 50–60% of respondents considering metrics like the number of open issues, number of total commits, and speed of responses (issue open and close dates and last commit dates) as important.

Figure 8: Trends in popularity features of an OSS component.

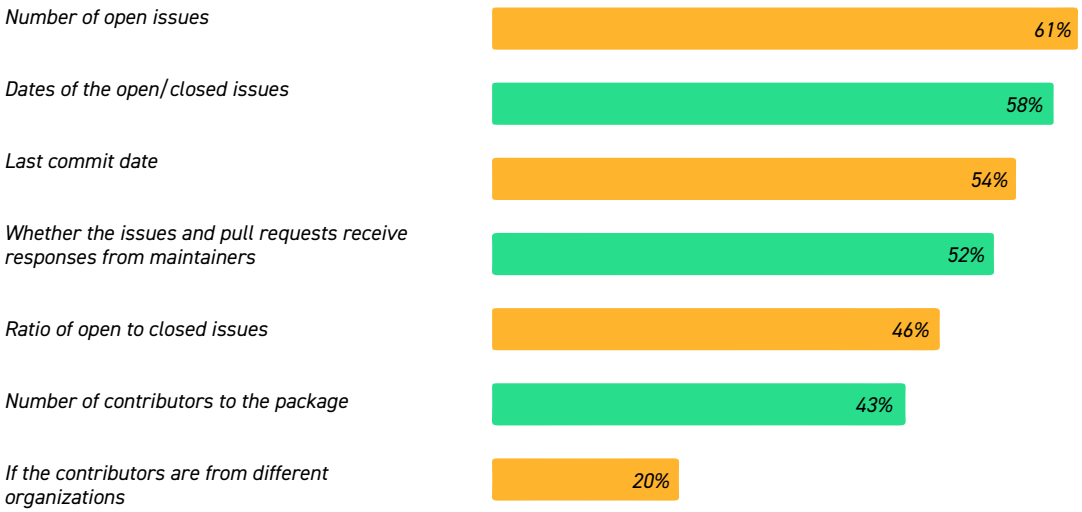


Drilling down further into attitudes towards issue resolution, we see that 50% of our respondents consider (in at least one way) how actively the code is maintained and how quickly bugs get resolved before they commit to the use of a component (Figure 9).

This shows that approximately half of the technical stakeholders are not concerned with active maintenance and constant bug fixing of the OSS project they're selecting. An actively maintained project is expected to be more secure since fixes are more likely to be pushed quickly and consistently to downstream users.[17] An abandoned OSS project will not receive any software updates. The number of contributors to a package and their organisational affiliations are less important for the evaluation and adoption of code, with only 43% and 20% of respondents actively looking at these metrics, respectively.

Developer approaches towards project maintenance point to two other less obvious outcomes. First, OSS users approach software as consumers of the code rather than members of a community building a useful feature together. Given that most stakeholders do not consider who is creating the software or how it came into being, but only that it has been used by other people and is thus trustworthy enough to use, it is reasonable to conclude that they have no intentions of contributing back to the package used. This is also supported by attitudes reported towards OSS community activity and documentation completeness. Second, only about 50% of the stakeholders surveyed actively examine the maintenance infrastructure of OSS components. This implies that about half of the stakeholders do not consider the possibility of introducing unintended bugs into their own projects and thus opening them up to software vulnerabilities. One possible reason could be that security thinking is passed on to a dedicated security team, as seen in the previous sections, and stakeholders are left with the role of perfecting the functionality of code.

Figure 9: Trends in maintenance features for OSS examination.



Takeaway:

80% of technical stakeholders look for project documentation, and 70% seek an active community around an OSS component. Only around 50% look for evidence in some form that the project is actively maintained. However, this search for documentation, community and reputable maintenance features is motivated by a consumer outlook towards open-source software that prioritises the ability to quickly resolve problems and treat the community as a 'help desk' rather than a group of people working together to create a useful service.

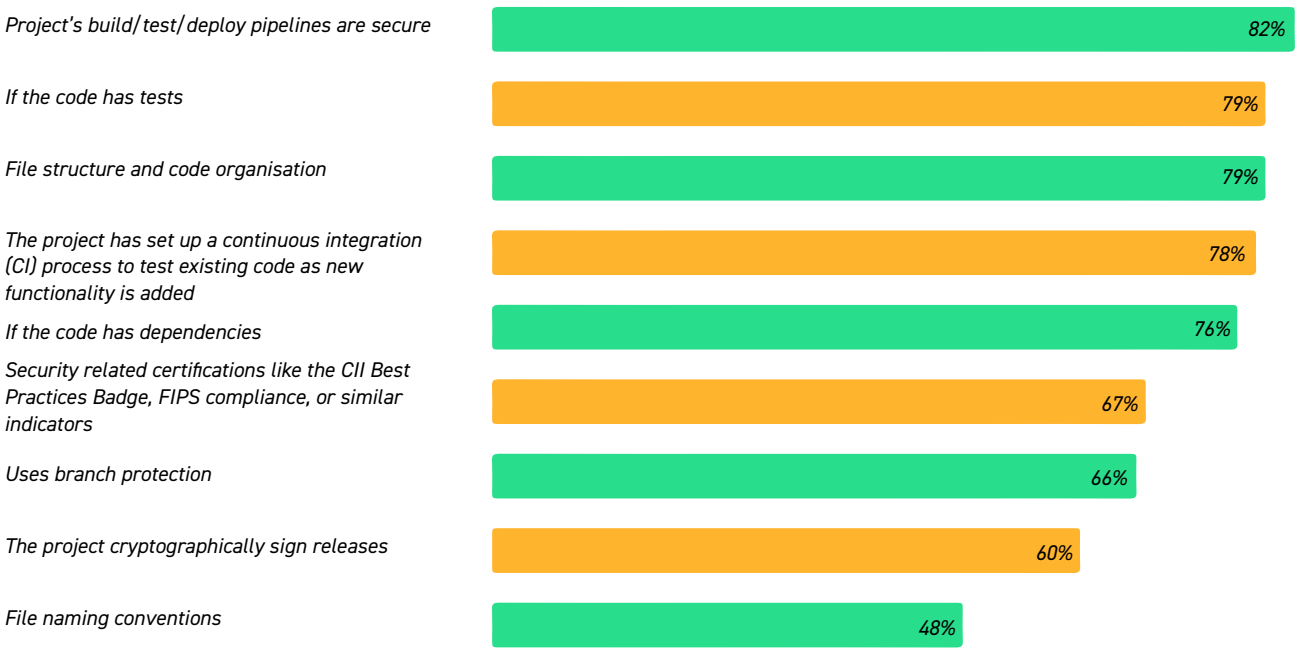
Adoption Checks

Adoption checks investigate the code itself once the potential software component has been shortlisted. This can include following best practices (Figure 10) such as conducting targeted testing of the software and its dependencies.

Some best practices are popular: testing the security of the project's build/test/deploy pipelines (over 80%), looking for the existence of a continuous integration process, and reviewing the file structure and code organisation (nearly 80%) to verify that the code functions correctly as new functionality is added. A lower percentage (though still more than 60%) of respondents stated that they ensure that the OSS component they are considering has security certifications or cryptographically signs releases.

Further research should explore the various ways in which this integrity testing is done since our sample also stated that participants rarely investigate software for security features once it has been approved for use by their organisation's security teams.

Figure 10: Best practices followed while assessing OSS components for use.



The most unexpected finding from our interviews was the prevalence of self-reported dependency verification and thoroughness of testing. Based on our literature review and media reports of security vulnerabilities affecting widely used software libraries, we expected to see much lower rates of dependency verification. Existing literature suggests that developers and other stakeholders do not fully comprehend the extent of direct or transitive dependencies that are present in the software they create. [18.19,20]

However, our findings indicate that 85% of respondents are aware of the dependencies in the software they select, and over 70% confirmed that the dependencies present can be validated (Figure 11).

Figure 11:
Trends in dependency verification.

The responses received on testing approaches agreed with the existing literature after we removed internally conflicting data (see the Limitations for more information). 65% of our sample said that they checked for the presence of tests. Meanwhile, checking for the presence of automated fuzzing tools was confirmed by 53% of the sample (Figure 12).

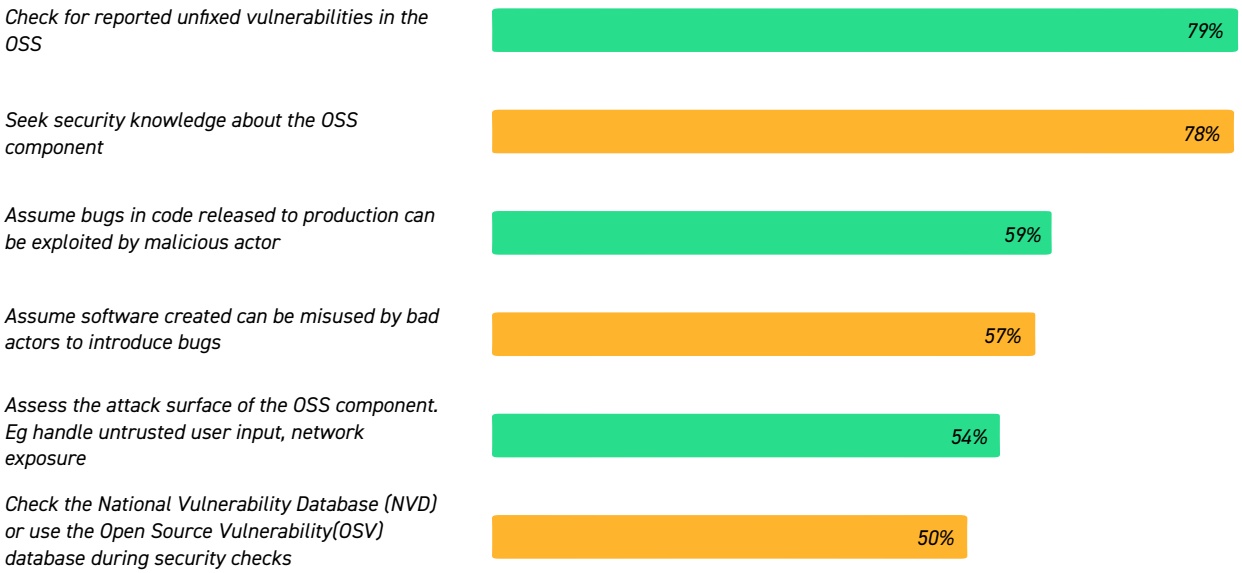
It is currently unclear whether this verification of tests and dependencies is done by the respondents themselves or is assumed to be done by security teams.

Figure 12: Software testing behaviours of the sample.

Further, nearly 80% of respondents test and seek general security knowledge of OSS components before committing to them. However, the assumption of targeted malice is much lower. Over 40% of the people in our sample do not think that the code they write can be misused or hacked by bad actors. Only half of the respondents check the vulnerability of databases to see if the tool they are considering is listed (Figure 13).

This corroborates the findings in the previous sections regarding the security team and issue maintenance. The security of the code does not seem to be the respondent's responsibility, and the possibility of bugs propagating through the software supply chain is not a primary concern for most employees. Extensively evaluating dependencies and testing are presumably more functionality-related if security is not seen to be a primary priority.

Figure 13: Respondent assumptions of threat in the projects they create



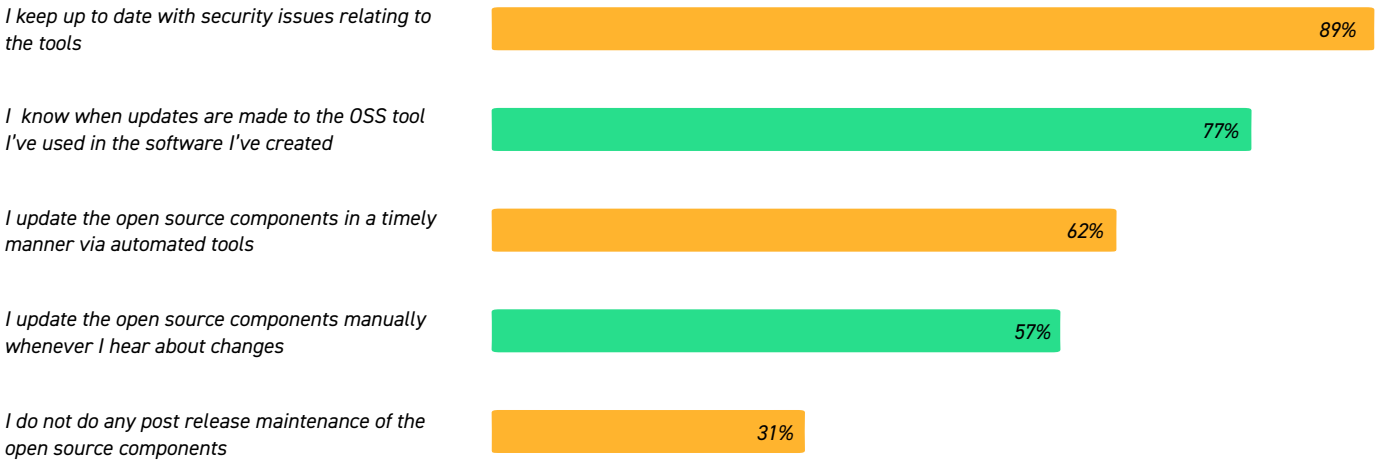
Takeaway:

Over 40% of people do not expect accidental exploitation of vulnerabilities or intentional malice from bad actors when they create software.

Post-Release Checks

The prevalence of post-release maintenance habits is high, with nearly 90% of respondents stating that they keep up to date with changing security information and other updates about the OSS after the product they have created is deployed to intended users. Only 30% of the respondents report not doing any post-release maintenance (Figure 14).

Figure 14: Respondent actions towards OSS once code has been released to production for use.



Takeaway:

70% of technical stakeholders carry out post-release maintenance of the OSS components they've used to create their projects.

03

Conclusions

We find that while the security of software is an important feature in the adoption of OSS components, it is still secondary to functionality. Many do not actively consider the threat of bad actors when they create software to be used by others. They do, however, still carry out post-release maintenance to ensure that the products they have created remain up to date.

Most stakeholders depend on security teams to thoroughly evaluate the security of the software they have selected for use. Once an OSS project has been approved for use by the in-house team, no further security checks are deemed necessary. Further, users approach OSS components with a consumer outlook. The search for documentation, community, and reputable maintenance features is motivated by a consumer outlook towards open-source software which prioritises the ability to get problems solved quickly and treats the community as a 'help desk' rather than a group of people working together to create a useful service.

Limitations

Our survey is based on self-reported data, so respondents are susceptible to providing answers that show them in a favourable light or which they think the interviewees want to hear. We have tried to protect against this limitation by rewording any questions that could lead the respondent towards a particular answer and by reminding them that the survey is not a reflection of their work ethic.

Subjective Analysis:

Reducing complex categories like reputation and community down to specific metrics necessitates a degree of subjective analysis. We understand that different researchers may have different opinions on the categorisation.

Defining "Open-Source Code":

During our pilots, we realised that the term "open-source code" meant different things to people from companies with different business functions. Some think of it as free software libraries that they can use without having to pay for them, while others only interact with it as an end product like Firefox.

Community vs Enterprise Software:

Employees using enterprise open-source software that is provided through companies like Red Hat depend on support from Red Hat to customise the components to business needs. This essentially creates a consulting-like relationship between users and maintainers of the software rather than there being a community that contributes to and builds each other's work. We learned this from our pilots, and it was repeated by many of our participants. Our research currently does not make a distinction between community open-source and enterprise open-source software; future research should explore this dynamic further.

Testing section Results:

There were inconsistencies in the results on testing (see Figure 3). It is impossible to have more people checking to see if the functionality was sufficiently covered than those who checked if tests were present. To deal with this contradiction, we removed the data points on the thoroughness of testing from our analysis and kept only the methods of testing.

Figure 3: Inconsistencies in the results on OSS testing.

Endnotes

1 Timothy B. Lee, "The Heartbleed Bug, Explained", Vox, 14 May 2015, vox.com

2 Eran Orzel, 2021 Software Supply Chain Security Report. Argon Security, January 2022. info.aquasec.com

3 Dina Temple-Raston, "A 'Worst Nightmare' Cyberattack: The Untold Story of The SolarWinds Hack", NPR, 16 April 2021, npr.org

4 "Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure", Ford Foundation, fordfoundation.org

5 "The State of Enterprise Open Source", Red Hat, redhat.com

6 Valentina Lenarduzzi, et al., "Open Source Software Evaluation, Selection, and Adoption: A Systematic Literature Review", 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), (2020): pp. 437-444, doi: 10.1109/SEAA51224.2020.00076.

7 Gene M. Alarcon et al., "Trust Perceptions of Metadata in Open-Source Software: The Role of Performance and Reputation", Systems 8, no. 3 (2020): 28, doi.org.

8 Shao-Fang Wen, Mazaher Kianpour, and Stewart Kowalski, "An Empirical Study of Security Culture in Open Source Software Communities", ASONAM '19: Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (2019): 863–870, doi.org

9 Oyvind Hauge, Thomas Osterlie, Carl-Fredrik Sorensen, and Marinela Gerea, "An Empirical Study on Selection of Open Source Software – Preliminary Results", 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (2009): 42–47, doi: 10.1109/FLOSS.2009.5071359.

10 Vieri del Bianco, Luigi Lavazza, Sandro Morasca, and Davide Taibi, "A Survey on Open Source Software Trustworthiness", IEEE Software 28, no. 5 (2011): 67–75, doi: 10.1109/MS.2011.93.

11 Vieri del Bianco, Luigi Lavazza, Sandro Morasca, Davide Taibi, and Davide Tosi, “An Investigation of the Users’ Perception of OSS Quality”, Open Source Software: New Horizons, IFIP Advances in Information and Communication Technology 319, (2010), doi.org

12 Shao-Fang Wen, “Software Security in Open Source Development: A Systematic Literature Review”, 2017 21st Conference of Open Innovations Association (FRUCT), (2017): 364–373, doi: 10.23919/FRUCT.2017.8250205.

13 “Explore”, OpenSAMM, accessed June 2022, opensamm.org

14 Gene M. Alarcon et al., “Trust Perceptions of Metadata in Open-Source Software: The Role of Performance and Reputation”, Systems 8, no. 3 (2020): 28, doi.org.

15 “Our Software Dependency Problem”, research!rsc, accessed June 2022 , research.swtch.com

16 Vieri del Bianco, Luigi Lavazza, Sandro Morasca, and Davide Taibi, “A Survey on Open Source Software Trustworthiness”, 67–75.

17 “Our Software Dependency Problem”, research!rsc.

18 “The Modern Packager’s Security Nightmare”, Michał Górny, accessed June 2022, blogs.gentoo.org

19 Ken Thompson, “Reflections on Trusting Trust”, Communications of the ACM 27, no. 8 (1984): 761–763, cs.cmu.edu.

20 “Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies”, Alex Birsan, Medium, accessed June 2022, medium.com.

Centre
for Internet
& Society