Open Source Software Best Practices and Supply Chain Risk Management

by Forward Digital

March 2024

Abstract

This report aims to map and evaluate existing best practices for managing and mitigating risks related to open-source software across various organisational contexts. It seeks to provide a comprehensive overview of the guidance offered by international governments, industry standards, and formal standards bodies, focusing on the usage, production, security, and licensing of open-source software and the management of software supply chain risks.

Contents

1 Executive Summary

This report outlines and evaluates effective strategies for managing and mitigating risks associated with open-source software (OSS) in various organisational contexts. It provides a detailed overview of guidance from a varied range of sources, including national governments, industry standards, existing practices, and formal standards bodies. The report ends with a set of recommended best practices for managing OSS and mitigating supply chain risks effectively. The best practices recommended in this report have been selected as those most appropriate for organisations of any size and sector.

The research methodology encompasses a broad literature review and insights gathered from industry practitioners' interviews. This dual approach enables a thorough analysis of current practices, challenges, and opportunities within OSS management, ensuring a range of perspectives are considered.

The research presents a broad and evolving view of OSS best practices, supported by a wealth of contributions from the community. However, it highlights a notable gap in industry-specific best practices and best practices suitable for organisations of different sizes. Due to the broad and evolving view of OSS best practices, our research found discrepancies and variations in approaches to OSS adoption, management, and community engagement. Large technology companies also play a critical role in setting standards and best practices, significantly impacting the ecosystem.

Based on our analysis of existing guidance and best practices, this paper proposes four best practices that constitute a proportionate and reasonable approach to OSS risk management that are also achievable for organisations of any size. The report recommends that organisations should be encouraged to do the following:

- Establish an internal OSS policy to manage the adoption of OSS components.
- Create a Software Bill of Materials (SBOM) to track OSS components and their dependencies.
- Continuously monitor the organisation's software supply chain using a software composition analysis (SCA) tool to identify vulnerabilities and licensing issues.
- Promote active engagement with the OSS community to attract new talent,

level the competitive playing field, foster innovation, improve reputation, and ensure high-quality OSS components and a sustainable OSS ecosystem.

Adopting tooling to automate OSS management processes is strongly recommended to alleviate time and resource constraints, particularly for smaller organisations. Tooling is a low-cost or free alternative to manually following the best practices outlined in this report. When using open source software components in software development, these actions are recommended for increasing the trustworthiness and maturity of OSS components and managing associated risks effectively.

In addition, this report recommends that future research and policy should focus on developing scale appropriate best practice guidance, encouraging organisations to contribute back to the OSS community, explore if industry-specific guidance is needed for managing OSS components, assessing the impacts of community engagement on OSS quality and security, and formulating standardised metrics for evaluating the maturity and trustworthiness of OSS components. Such efforts would further enhance the ability of organisations to manage OSS risks and capitalise on the opportunities OSS presents effectively.

2 Introduction

As open-source software (OSS) grows in popularity, reliance on it as a cornerstone to productivity and innovation grows. With this growth, the need to consider and address risks associated with OSS adoption also grows. From strategic organisational choices concerning adopting and managing OSS to the everyday technical considerations involving licensing, tooling, and community engagement.

The use of OSS has been on a consistent upward trajectory, with each year surpassing the last in terms of adoption rates. In 2023, OpenLogic reported an 80% increase in OSS adoption over the figures reported in 2022 (OpenLogic, 2023). With an increase in the adoption and management of OSS, there is now, more than ever, a need to understand the best practices behind the adoption, management, licensing, and tooling associated with OSS.

This report, commissioned by the Department for Science, Innovation and Technology (DSIT), seeks to analyse the landscape of industry practices for using OSS and managing OSS risks. Special attention was given to organisations developing software products and services for business to business (B2B) sales and or-

ganisations delivering B2B services/products that develop software for internal processes or for providing their services/products. It aims to identify, evaluate, and recommend best practices organisations can implement to ensure a robust and secure OSS management strategy. The best practices focus on the adoption, management, and community aspects of OSS, as well as the organisational context in which OSS is used.

Using these best practices, organisations can safeguard against licensing and Intellectual Property (IP) risks, allow only trusted and mature dependencies into their software supply chain, and map a clear picture of their own OSS supply chain.

This report begins with a literature review, examining academic papers, best practices, case studies, industry standards, and a series of interviews with field experts. Each piece of "knowledge" from the literature review was evaluated for its comprehensiveness across four sub-categories of OSS management: adoption, management, community, and context.

Here, we present the findings of our research, which are divided into two parts. The first part maps and evaluates existing guidance on OSS adoption, management, and community. The second part sets out our recommended best practices for organisations to adopt, underpinned by the findings of the literature review. Our best practice list aims to be both robust and flexible, allowing organisations to tailor their OSS management strategy to their specific needs and context.

3 Methodology

This section outlines the approach adopted to collate and scrutinise the existing guidance and best practices pertinent to open-source software (OSS) management.

The methodology encompasses a comprehensive literature review, interviews with industry experts, analysis of case studies, and an examination of formal standards to ensure a robust and wide-ranging understanding of OSS practices.

3.1 Academic Papers

3.1.1 Database Selection

The selection of databases for academic papers included IEEE Xplore, ACM Digital Library, ScienceDirect, JSTOR, Google Scholar, and ProQuest to ensure comprehensive subject coverage.

3.1.2 Search Terms

A strategic search was conducted using keywords such as “open source software”, “OSS best practices”, “OSS case studies”, “industry standards for OSS”, “OSS development”, and “OSS project management”. Boolean operators were used to refine the searches, for example, “open source software” AND “best practices” OR “case studies”.

3.1.3 Criteria for Inclusion and Exclusion

The selection of academic papers was based on the following criteria:

Publication Date Emphasis was placed on literature published within the last five years to ensure the inclusion of the latest developments and insights.

Relevance Only materials relevant to key OSS topics, such as open-source adoption, open-source management, and open-source community engagement, were selected.

Quality of Research Preference was given to detailed studies with significant findings or data, evaluated based on their research quality and contribution to the body of OSS knowledge.

Variety of Perspectives A wide range of perspectives from around the world was sought to ensure a diverse and comprehensive understanding of the subject.

3.2 Best Practices

3.2.1 Source Selection

Industry Publications and White Papers Materials from well-known OSS organisations and tech companies were focused on. Examples include the Linux Foundation, Apache Software Foundation, and GitHub's OSS reports.

Online Blogs Blogs from leading technology companies and OSS communities were reviewed.

Policy Reports Reports from government bodies and international organisations, such as the United Kingdom, the European Union, the United Nations, and the United States Department of Defense, were explored.

3.2.2 Search Terms

Keywords like "OSS adoption best practices", "OSS management strategies", "OSS community engagement", and "OSS licensing" were used to find relevant materials.

3.2.3 Criteria for Inclusion and Exclusion

Best practices were judged based on the following criteria:

Adoption Best practices that showed a comprehensive approach to adopting OSS, including tools, strategies, and risk management, were chosen.

Management Focus was on best practices that showed an extensive and well-thought-out strategy for managing OSS projects, including dependency management, security, and continuous monitoring.

Licensing and Compliance Included best practices that offered clear guidance on licensing and compliance issues, including how to handle licensing risks and ensure compliance with OSS licenses.

Community Engagement Favoured best practices that offered advice on how to engage with OSS communities, including how to contribute to OSS projects and manage community feedback.

Real-world Relevance Focused on best practices proven to work in organisations adopting and managing OSS.

Proven Success Preference was given to practices with documented success stories or positive outcomes.

3.3 Case Studies

3.3.1 Source Selection

Open Source Bodies Case studies from leading OSS organisations were selected.

Tech and Business Publications Case studies from reputable tech and business publications, such as Wired, TechCrunch, and Harvard Business Review, were reviewed.

3.3.2 Search Terms

Keywords like "OSS case studies", "OSS success stories", "OSS challenges", and "OSS implementation" were used to find relevant materials.

3.3.3 Criteria for Inclusion and Exclusion

Varied Industries and Sizes Case studies were chosen from different fields such as technology, healthcare, and education, and included both small startups and large corporations to ensure a wide range of OSS applications.

Successes and Lessons A mix of positive examples and those offering warnings was compiled to learn from both mistakes and successes.

International Examples Case studies from around the world were selected to observe how different regions approach OSS.

Recent Studies The focus was on the latest case studies (since 2019), but key older examples that have significantly influenced OSS use were also included.

Direct Relevance Case studies were ensured to be directly related to key OSS issues such as development, management, or community involvement.

Actionable Lessons Case studies that provide clear takeaways or advice for others using OSS were favoured.

Reliable Sources The reputation of the case study's publisher was verified to ensure the information is trustworthy.

Varied Views The case studies were ensured to cover a range of perspectives, including those from different roles within the OSS project and external viewpoints.

3.4 Industry Standards

3.4.1 Source Selection

Established Organisations Standards set by well-known groups such as the Open Source Initiative (OSI), International Organization for Standardization (ISO), and Internet Engineering Task Force (IETF) were sought.

Legal and Compliance Standards Key legal and compliance guidelines affecting OSS were identified, including licensing laws, security standards, the General Data Protection Regulation (GDPR) for data protection, and accessibility rules.

Community Guidelines Practices developed within OSS communities, such as coding conventions, governance rules, and ways to collaborate, were considered.

3.4.2 Search Terms

Keywords like "OSS standards", "OSS legal guidelines", "OSS security standards", and "OSS community guidelines" were used to find relevant materials.

3.4.3 Criteria for Inclusion and Exclusion

Standards were chosen based on the following criteria:

Scope and Detail The thoroughness with which the standards cover OSS needs, from creation and sharing to licensing and upkeep, was reviewed.

Relevance The extent to which these standards are adopted among developers and across the software industry overall was examined.

Adaptability Whether the standards can support differing organisations, both large and small, was checked.

Comprehensiveness Standards that cover a wide range of OSS topics, from licensing to security, were favoured.

3.5 Interviews

3.5.1 Selection of Experts

We engaged with developers, project managers, DevOps developers, and C-suite level executives to get a broad view of OSS within different organisations. Our total interview count was eight.

To find a broad range of interviewee, from varying backgrounds and sectors, we connected with people on public technology-focused forums like Sheffield Digital, techlondon, and Berlin Techs. Identified potential candidates through LinkedIn, using keywords like “open-source”, “OSS”, and “OSS management”. Reached out to people that had written articles or blog posts about open-source. Finally, we used our own personal networks to find potential interviewees.

All interviewees were offered a £100 charitable donation on their behalf to incentivise participation.

3.5.2 Preparation

Key Topics We focused on the interviewees’ experiences with OSS, including their approach to adopting new OSS, managing OSS projects, and engaging with OSS communities.

Open-ended Questions We asked open-ended questions to encourage detailed responses and insights.

Consent and Recording We secured consent from the interviewees and recorded the interviews for accurate data analysis.

3.5.3 Interview Agenda

1. **Introduction:** The session began with the research team introducing themselves and explaining the purpose of the interview. Subsequently, the interviewees introduced themselves and described their careers up to the present day.
2. **Project Management:** The discussion covered how the interviewees manage OSS dependencies, focusing on the processes of adoption, integration, and maintenance.
3. **Risk Management and Security:** The conversation explored the interviewees' approaches to handling security and compliance issues associated with OSS. Additionally, it delved into any processes they have established to manage crises related to OSS.
4. **Licensing and Tooling:** The interviewees were asked about their strategies for OSS licensing and the tools they utilise to manage OSS projects, vulnerabilities, and dependency management effectively.
5. **Community Engagement:** This part of the interview discussed how the interviewees engage with OSS communities, including their or their organisation's contributions to OSS projects and how they handle feedback from the community.
6. **Successes and Challenges:** The interviewees shared their experiences with OSS, detailing any challenges they have encountered and the strategies employed to overcome them, as well as highlighting successes they have achieved with OSS.
7. **Closing Remarks:** The interview concluded with the interviewees being given an opportunity to share any final thoughts on the lack of consensus, critical factors for successful OSS implementation, and their vision for the future of OSS within their organisation in the context of the wider industry.

3.5.4 Data Analysis

Transcription The interviews were recorded and Artificial Intelligence (AI) was employed to transcribe them, ensuring the accuracy of the data analysis.

Thematic Analysis

- Recurring themes and patterns were identified across the interviews.
- The responses were categorised into three sub-categories of OSS management: adoption, management, and community engagement.
- An analysis of the responses was conducted to identify commonalities and differences in the experiences and practices of the interviewees.

4 Mapping and Evaluation of Existing Guidance

To summarise our findings, we have grouped all existing guidance into three categories: adoption, management, and community.

Adoption The process and strategy an organisation employs to integrate open-source software (OSS) into its operations and projects.

Management How an organisation oversees and maintains its open source software usage, including updates, security, and compliance.

Community The extent to, and manner in which, an organisation engages with and contributes to the wider open source community.

4.1 Adoption

Open-source adoption includes the strategic and tactical considerations of introducing open-source software to your organisation. Having a robust adoption process is critical to ensuring that new open-source components are correctly evaluated before being integrated into the organisation's systems.

In this section, we will discuss the best practices for adopting open-source software, including the internal considerations organisations must make, the metrics

by which to judge the quality of OSS components, the approval process, and the licensing of OSS components.

4.1.1 OSS Policy

An OSS policy is a formal document that outlines the rules and guidelines for using open-source software within an organisation.

Our literature review found an OSS policy to be a recurring theme in the best practices for adopting open-source software. Best practices recommend that organisations have a formal OSS policy in place.

An OSS policy should (Hammond, 2009);

- Be concise
- Developer consumable
- Involve general counsel early and often
- Classify OSS license types
- Take control of the process
- Measure internalisation
- Be revisited and revised on a regular basis

It should include the goals for OSS adoption, the acquisition process, the rules for the business case, and guidelines for the appropriate use of OSS (Hammond, 2009).

Lacking a formal policy for OSS usage and adoption, which typically includes licencing guidelines, can lead to legal and security risks (Schofield, 2008).

However, a policy alone is not enough. It is essential to ensure that all employees understand and follow the policy. Most importantly, the policy steps should apply to the organisational context (Osborne et al., 2023).

Open-source policies are a formal way for an organisation to set guidance and rules for using open-source software. As we will see later, most developers use their own discretion rather than a formal organisational process, which can lead to inconsistencies and licensing risks.

A policy is an excellent way for an organisation to ensure that all developers are following the same process, the correct stakeholders are included in the adoption process, and that the process is tailored to the organisation's needs.

4.1.2 Maturity Models

Maturity models are a way to evaluate the quality of open-source software components. They can play a key role in the adoption process, as they allow for evaluating OSS components before they are integrated into the organisation's systems. We found that there are several maturity models for evaluating OSS components; however, there wasn't a consensus on which model was the best.

Fundamentally, the adoption of OSS components differs from the adoption of proprietary software due to the fact OSS is stored in public repositories, accessible by anyone, and is often developed by a community of developers (Umm-e-Laila et al., 2016). Due to this, the quality of open-source software can be very different from one product to another; therefore, evaluation methods have been developed to evaluate OSS components properly.

The most popular OSS evaluation methods are the Capgemini-Open Source Maturity Model (C-OSSM), Navicasoft-Open Source Maturity Model (N-OSSM), Qualification and Selection of Open Source (QSOS), Open Business Readiness Rating (Open BRR) and Easiest Open Source (E-OSS) (Umm-e-Laila et al., 2016).

None of those interviewed were familiar with these models, let alone using them in their organisation. This suggests that these models are not widely used in practice. We also found, from our literature review, that there is no consensus on which model is the best, with Umm-e-Laila et al., 2016 going as far as to say that "in order to take advantage of OSS properly, it is recommended to propose a new framework/model that will eliminate the weakness of all models.".

We can conclude that while maturity models are an excellent way to evaluate OSS components, there is no consensus on which model is the best, and they are not widely used in practice. From our interviews, we found that most developers judge the maturity of OSS components based on their own discretion, usually stemming from their own experiences and the experiences of their colleagues.

4.1.3 Trust in Open-Source Software

Trust in OSS is a critical concept when adopting OSS components. How does one come to trust an OSS component? More often than not, "there is no sound basis for trust in the Software Ecosystems (SECO) hubs", with trust being considered "founded or unfounded" (Hou et al., 2022).

Trust is subjective and based on the end-developer's perception and intrinsic and extrinsic motivations. When adopting a new OSS component, the developer will consider intrinsic factors such as:

- Source code
- Reliability
- Security

Extrinsic factors such as:

- Documentation
- Structural assistance
- Cost

Quality is the most discussed factor in trustworthiness, with most developers considering the quality of the OSS component as the most critical factor in trustworthiness (Hou et al., 2022).

From our literature review and the interviews, we found that each developer uses their own trust model and that there is no documented process for evaluating the trustworthiness of OSS components within an organisation.

Outside of academic papers, trustworthiness wasn't mentioned in any of the best practices we reviewed.

This is a significant gap in the best practices landscape, as trust plays a vital role in adopting OSS components.

4.1.4 Approval process

The approval process is a formal process for evaluating and approving OSS components before they are integrated into the organisation's systems. It's an opportunity for the organisation to evaluate the OSS component and ensure that it meets

the organisation's needs and standards, and engage relevant stakeholders in the approval process.

When adopting OSS, there are several "deal-breakers" that can prevent the adoption of an OSS component. Outlined by Spinellis, 2019, these include:

- Lacking functionality
- Incompatible license
- Nonfunctional properties

Businesses should approve OSS based on technical needs, licensing, longevity, and business factors like cost and benefits. Yet, businesses must balance risk with commercial advantages and be willing to adapt their methods quickly. The approval process should be formal, with a clear set of guidelines and rules, and should be managed by a central team and engage relevant stakeholders (Butler et al., 2022).

When selecting an OSS component, ensure it demonstrates the following security and maintenance best practices (Department of Defense, 2022):

- **Detection Tool Usage:** Actively employs detection tools within its integration pipeline and externally.
- **Vulnerability Reporting Transparency:** Has a clear process for the open reporting and documentation of security vulnerabilities.
- **Security Review History:** Possesses a record of security reviews.
- **Cybersecurity Testing:** Undergoes regular cybersecurity assessments, notably third-party audits, to verify security measures.
- **Issue Resolution:** Demonstrates a commitment to promptly addressing and resolving reported issues.
- **Timely Vulnerability Remediation:** Shows a track record of quickly remedying vulnerabilities.

We found that an approval process was a recurring theme in the best practices for adopting open-source software. Every interviewee demonstrated some approval process for the adoption of OSS components. Smaller companies tended to build a minimal viable product (MVP) with the OSS in question; larger teams had a multistep process that involved multiple stakeholders.

The US Department of Defense, 2022 recommendations primarily emphasise the quantifiable attributes of OSS dependencies, including security features and measurable performance metrics. In contrast, Butler et al., 2022 analysis portrays the approval process as somewhat ambiguous, describing it as more reliant on assessment and discretion than on strict, predefined criteria.

To conclude, the approval process for adopting open-source software is dynamic and customised to each organisation's specific requirements, yet it remains a universally acknowledged standard that all entities conform to.

4.1.5 Software Bill of Materials

Software Bill of Materials (SBOM) is a formal list of components used in a software system. It allows for tracking OSS components and their dependencies, which is essential for managing the security and compliance risks associated with OSS.

Since 2018, SBOMs have surged in popularity, driven in part by a collaborative effort headed up by the American National Telecommunications and Information Administration's (NTIA) multi-stakeholder process (National Telecommunications and Information Administration, n.d.). This work culminated in the US government's issuance of an executive order mandating the use of SBOMs in all federal software (The White House, n.d.) in direct response to the SolarWinds supply chain attack (Schwartz, 2021).

The two primary standards for SBOM are SPDX and CycloneDX.

SPDX17 is an open standard developed by the Linux Foundation to communicate details of a SBOM, including components, licenses, copyrights, and security references, recognised internationally as ISO/IEC 5962:202118 (*System Package Data Exchange (SPDX®)* 2024). CycloneDX19, originating from the Open Web Application Security Project (OWASP) community, is an SBOM standard crafted for application security and supply chain component analysis, now extended to encompass a broader array of applications such as software-as-a-service BOM (SaaSBOM) (*CycloneDX* 2024).

Both formats are popular, yet organisations are advised to select the format that most aptly meets their specific requirements (Alvarenga, 2023a).

At a minimum, an SBOM must contain three categories (United States Department of Commerce, 2021);

Data fields Data fields must be standardised and contain essential information on each component for effective tracking and management. The aim is to ensure these components can be easily identified throughout the software supply chain, linking them to other valuable data sources (United States Department of Commerce, 2021). Naming conventions should try and follow the guidance outlined by NTIA Multistakeholder Process on Software Component Transparency Framing Working Group, 2021.

Automation Support Automation support is essential for the effective use of SBOMs, as it allows for the automatic generation and updating of SBOMs, which is essential for managing the large number of OSS components used in modern software systems.

The data formats that are being used to generate and consume SBOMs are:

- Software Package Data eXchange (SPDX)
- CycloneDX13
- Software Identification (SWID) tags

Practices and Processes Practices and processes are essential for the effective use of SBOMs, as they allow for the effective management of OSS components and their dependencies, necessary for managing the security and compliance risks associated with OSS.

There are two elements that should be explicitly stated in the SBOM:

Frequency – When a software component is updated or new information about its components is discovered, the supplier must issue an updated SBOM to reflect these changes (United States Department of Commerce, 2021).

Depth – An SBOM must include all primary components and their transitive dependencies, ensuring top-level dependencies are detailed enough to identify all subsequent dependencies recursively (United States Department of Commerce, 2021).

4.1.6 Licencing

The licensing of OSS components is a crucial consideration when adopting OSS as it can significantly impact the organisation's ability to use the OSS component

and considerably affect the organisation's ability to distribute the software that uses the OSS component.

As of 2022, 78% of open source components are under permissive licenses, which allow users wide freedom to use, modify, and distribute the software, with minimal requirements on how it can be redistributed. Meanwhile, copyleft licenses, which require that any modified versions of the software also be distributed under the same license terms, make up the remaining 22% of open source components (Murray, 2022).

The Apache License is the most popular OSS license, with 30% of all OSS projects using it (Murray, 2022).

Larger organisations (>5000 employees) are more likely to have an internal legal team familiar with OSS licensing when compared to small organisations (<100 employees) 31.46% vs 22.30%, respectively (OpenLogic, 2023).

To help reduce the risk of legal problems arising with the use of open-source software (Mathpati, 2023), organisations should watch over all open-source code being brought into the organisation, ensuring that licence requirements are met. To achieve this, there are several tools available, such as (The Linux Foundation, 2024):

- **Black Duck Protex** – A fee-based tool for license compliance and open source management.
- **Copyright review tools** – Command line utilities for copyright file management.
- **FOSSA** – Automates code dependency tracking and license compliance.
- **FOSSology** – An open-source toolkit from the Linux Foundation for license compliance featuring a web UI for compliance workflows.

Managing the licensing of OSS components is a key consideration when adopting OSS. Large organisations often have the advantage of an internal legal team to navigate open-source software (OSS) licensing, a resource that smaller organisations lack. Therefore, relying on internal legal teams for OSS license management cannot be deemed a universally applicable best practice, as it disadvantages smaller entities. The easiest way to manage your OSS licensing is to use a tool that can automatically track and manage the licensing of OSS components, as outlined by The Linux Foundation, 2024.

4.2 Management

4.2.1 Regular vulnerability assessments

Regular vulnerability assessments identify, prioritise, and apply patches to systems and software to address vulnerabilities. As the OSS landscape constantly evolves, organisations encounter thousands of new vulnerabilities. Yet, many businesses do not have a robust patch management strategy in place and fail to implement critical patches promptly, leaving them vulnerable to security breaches (Hackerone, 2024).

With an estimated 84% of open source components having at least one vulnerability (Sawers, 2021), regular vulnerability assessments are essential for managing the security risks associated with OSS.

A rough overview of the process is as follows (Hackerone, 2024):

1. **Initial Preparation** – Defining the scope and goals of the vulnerability testing to ensure a targeted and practical assessment.
2. **Vulnerability Testing** – Running automated tests to identify vulnerabilities within the systems included in the predefined scope.
3. **Prioritise Vulnerabilities** – Assessing which vulnerabilities are most critical, requiring immediate attention, and evaluating their potential business impact.
4. **Create Vulnerability Assessment Report** – Produce a detailed report outlining medium and high-priority vulnerabilities found and recommended remediation strategies.
5. **Continuous Vulnerability Assessment** – Conducting scans for vulnerabilities on a continuous basis to verify if previous vulnerabilities have been effectively remediated and to discover new ones.

Regular vulnerability assessments are frequently mentioned in the best practices for managing the security risks associated with OSS. Without automated vulnerability assessments, it is difficult to keep up with the large number of OSS components used in modern software systems and the many vulnerabilities discovered each year. Therefore, with automation, regular vulnerability assessments are a best practice for managing the security risks associated with OSS.

4.2.2 Tooling

Tooling is the functionality that allows for automating the management of OSS components. It can be very useful as a way to manage the large number of OSS components used in modern software systems without introducing unnecessary manual work. Any software project that actively works to reduce security vulnerabilities is less risky. Therefore, tools that aid in identifying security vulnerabilities within software projects are critical (Department of Defense, 2022).

There are three tools found from our literature review that are considered "leaders"; Snyk, Sonatype, and Synopsys (Black Duck) (Worthington et al., 2023). These tools are designed to help organisations manage the security risks associated with OSS. They can be used to automate vulnerability assessments, manage licensing, enforce an OSS policy, and generate SBOMs. With this in mind, due to the diversity of OSS components, there is no one-size-fits-all solution (Wen et al., 2019).

We consider tooling to be a best practice for managing the security risks associated with OSS. It enables organisations of all sizes to manage many OSS components used in modern software systems without introducing unnecessary manual work. There is a lot of research into tooling, and a wide range of tools are available. We found that many tool-promoting papers and articles are written by the companies that produce the tools, so it is down to the organisation to evaluate the tools and decide which one is best for them.

4.2.3 Continuous monitoring

Modern software applications are built using a complex web of dependencies, including open-source libraries and frameworks. This process is typically recursive, with each dependency having its dependencies, and so on (Dietrich et al., 2023). Infamously, this web of dependencies can lead to security vulnerabilities, as seen in the equifax data breach (Fruhlinger, 2020) and the log4j vulnerability (Gallo, 2022).

Continuous monitoring is the process of monitoring OSS components for vulnerabilities and other security risks on an ongoing basis. These can be manual but are often automated and part of an organisation's CI/CD pipeline. It is crucial for application developers to detect dependencies on vulnerable libraries as soon as possible, to assess their impact precisely, and to mitigate any potential risk (Ponta et al., 2020).

Now more than ever, developers are using automation to secure their dependencies; in 2023, developers merged 60% more pull requests from GitHub's Dependabot, a tool that automates dependency updates, than in 2022 (Daigle, 2023).

Software composition analysis (SCA) can be used to conduct regular vulnerability assessments, can be implemented as part of the continuous integration/continuous deployment (CI/CD) pipeline, and can be used to enforce an OSS policy (Alvarenga, 2023a). SCAs will provide an inventory of all open-source components used in a project, including their versions and licenses, and will also identify any known vulnerabilities in these components; effectively, producing an SBOM (Alvarenga, 2023a).

Due to the dynamic nature of open-source, continuous monitoring is essential for managing the security risks associated with OSS. An SCA is a great way to monitor OSS components continuously and can be implemented as part of the CI/CD pipeline and can be used to enforce an OSS policy. It is a best practice for managing the security risks associated with OSS.

4.2.4 Binary repositories

A binary repository manager is a tool designed to store and manage binary files (the compiled version of your source code) and their metadata. Some popular binary repository managers include JFrog Artifactory, Sonatype Nexus Repository, and GitHub Packages.

A binary repository manager is indispensable for managing open-source components effectively. It facilitates the caching of local copies of these elements, ensuring that frequently used packages remain accessible even during downtimes of external repositories. This capability is essential for maintaining continuous project development and operations without interruptions.

Moreover, such a tool distinguishes between approved third-party artefacts and those pending approval, significantly improving the management and visibility of open-source components within projects. It also allows access to specific libraries to be excluded and limited, thereby safeguarding projects from incorporating non-compliant artefacts (Wainstein, 2018).

By caching versions of all open-source software components, you also safeguard against incidents like the left-pad debacle, where the withdrawal of a minor package from the npm package manager (a package manager for the JavaScript pro-

gramming language) resulted in widespread failure in numerous projects (Williams, 2016).

There is a licensing concern, specifically with GPL-licensed open-source software. The GPL license requires that any software distributed that includes or is derived from GPL-licensed code must itself be available under the GPL. If a binary repository contains GPL software and is shared externally, the entire repository may need to be open-sourced. For example, Microsoft released Hyper-V drivers with GPL-licensed binaries, leading to Microsoft having to open-source the drivers in question (Linux Network Plumber, 2009).

Binary repositories appear to be a polarising issue in the literature. The Department of Defense, 2022 recommended binary repositories as a best practice for managing the security risks associated with OSS, whereas Anand, 2023 (Salesforce) mentioned securing a repository as a best practice, but didn't mention binary repositories specifically.

4.2.5 Risk management and mitigation

Risk management and mitigation is the process of identifying, assessing, and prioritising risks, and then taking steps to reduce or eliminate them.

According to the Open Worldwide Application Security Project (OWASP), the most significant risk associated with open source software is "Vulnerable and Outdated Components" and has moved from 9th to the 6th most critical risk to web applications (OWASP, 2020).

OSS vulnerabilities grew by 50% year-on-year — from just over 4,000 in 2018 to over 6,000 in 2019 (Zorz, 2020).

One reason for the increase in vulnerabilities is the growing number of open source components being used in software projects from developers utilising the speed and convenience of open source, especially under time pressure, without considering the security implications. This highlights the need for formal processes to manage the risks. Typically, these correlated to implementing some continuous monitoring and regular vulnerability assessments, as outlined in 4.2.1 and 4.2.3 (Anand, 2023, Chandler et al., 2022, Mathpati, 2023). Such efforts should avoid being overly laborious and manual to ensure they do not frustrate developers and slow down the development process (Contrast Security, 2020).

Like those outlined in this report, the most effective way to mitigate OSS risks is

to follow best practices (Chandler et al., 2022).

Interestingly, our interviews and case studies found that the approach to risk and risk mitigation varied from organisation to organisation. Larger organisations seemed more proactive in mitigating risk, with dedicated teams and processes in place. Whereas smaller organisations were more reactive, only taking action when a risk materialised and looking to improve their processes after the event. This suggests there is a gap in the best practices landscape, as it doesn't consider the different approaches to risk management and mitigation based on the size of the organisation.

4.2.6 Static application security testing

Static application security testing (SAST) is a method of testing the security of an application by examining its source code, byte code, or binary code for security vulnerabilities.

Frequent scanning and security assessment of your repository aids in detecting security concerns at early stages, minimising the chance of significant security problems developing later on. Static code analysis tools can pinpoint typical security weaknesses, and dependency vulnerability scanners can highlight any recognised problems in your project's dependencies (Anand, 2023). SAST goes hand in hand with the continuous monitoring of OSS components, as it allows for detecting vulnerabilities in OSS components as soon as possible, allowing for their impact to be assessed and mitigated, see 4.2.3.

From our research, SAST is only suggested by Salesforce as a best practice. However, current tooling is comprehensive and can be used to automate the continuous monitoring of OSS components, as outlined in 4.2.2. Therefore, SAST can be considered a technical implementation of continuous monitoring.

4.2.7 Forking

Forking is the process of creating a new project based on an existing project. It allows you to monitor alterations in open-source components, as there will always be a connection to the original repository. One of the key advantages mentioned in the definition of open-source is the explicit permission granted to duplicate and independently alter the source code of an open-source project as desired (*The Open Source Definition* 2024).

However, it's important to be mindful that opting to create a private fork entails the responsibility of integrating any updates from the upstream version of the component. This responsibility grows as the differences between the forked components and the original increase. An ideal situation for forking is when you take an open-source component from a project that is unlikely to receive many updates in the future (Wainstein, 2018).

The process often includes creating a distinct OSS developer community (OSS-DC) for the forked version, allowing the organisation to focus on modifying or enhancing specific components of the software—usually those deemed critical for their purposes (Méndez-Tapia et al., 2021).

Forking does get mentioned in varying mediums as a best practice. However, the added responsibility of integrating updates from the upstream version of the component is often overlooked, and from our interviews, we found that forking is not widely used in practice.

4.3 Community

4.3.1 Open Source Program Office

An Open Source Program Office (OSPO) is a corporate entity that is responsible for managing an organisation's open-source efforts. The OSPO is a "central nervous system" for open source within an organisation and provides governance, oversight, and support for all things related to open source (Haddad, 2022). It trains staff, promotes efficient software development with open-source parts, guides open-sourcing projects, ensures legal compliance, and builds community engagement.

An OSPO helps bridge the gap between an organisation's internal development and the external open-source community, helping to ensure that the organisation is a good steward of open-source software and can reap the benefits of open-source adoption, all the while minimising risks (TODO Group, 2023).

Six common characteristics of successful open source programs (Walker, 2016) are:

- Value marketing and branding highly.
- Choose open-source communities that match your tech goals.

- Get good legal advice to balance risk and innovation.
- Your open-source efforts should support your product goals.
- Explain your support plans clearly to everyone involved.
- Hire someone passionate about open source to lead.

Uber, for example, was motivated to create its own OSPO because of the need to streamline support for various open-source initiatives. For instance, there was a significant demand for advice on incorporating open-source technology into external products from engineers. Key concerns included navigating compliance, licensing, and related legal matters. Additionally, there was a crucial need to offer engineers direction on how to release Uber's own software as open source, whether by initiating new projects or contributing to existing ones (*Uber Case Study* n.d.).

"It was natural and organic for Uber to create an open source program office since it allowed us to build our platform and scale the technology at unprecedented speed [...] in essence, Uber loves open source because it's essential to our success." (Hsieh, n.d.).

Since the OSPO is a relatively new concept, with Google opening one of the first OSPOs in 2004 (Google, 2022), the literature references mostly large corporate entities with the resources to create an OSPO. The Linux Foundation recommends a minimum of five employees to start a successful OSPO (Haddad, 2022). Therefore, an OSPO is not best suited for smaller organisations that cannot afford to dedicate the resources to an OSPO.

4.3.2 Open Sourcing Previously Proprietary Software

Open-sourcing previously proprietary software is the process of releasing software to the open-source community for free use and modification. This process can be beneficial for organisations; it allows for increased innovation, collaboration, and community engagement and can lead to the development of features that are beneficial to the organisation, as well as the resolution of issues that impact the organisation's operations.

There are a series of examples of organisations open-sourcing previously proprietary software. Some of the most notable examples include:

Microsoft Open-sourcing .NET By open-sourcing .NET, Microsoft allowed for a single, collaborative codebase across platforms rather than separate implementations. Ultimately, open development allows more community engagement to provide feedback and contributions, making the stack better for all (Landwerth, 2014).

Microsoft Open-sourcing Powershell Microsoft open-sourced PowerShell to make it more widely available and helpful for system administrators and developers who work with multiple operating systems. By making it open source and available cross-platform, more people can use PowerShell to automate tasks and manage their diverse computing environments that include different OSes (Foley, 2016).

NVIDIA Open-sourcing PhysX NVIDIA open-sourced PhysX because physics simulation has "dovetails" with AI, robotics, and computer vision. NVIDIA considered physics simulation "foundational for so many things" and open-sourcing it would allow it to be developed and applied more widely than NVIDIA could do alone (Lebaredian, 2018).

Microsoft Open-sourcing Live Writer Microsoft open-sourced Live Writer to allow the community to continue to develop and improve it, as Microsoft no longer had the resources to maintain it (ARS STAFF, 2015).

The primary motivation for open-sourcing previously proprietary software is to allow for increased community engagement, feedback, and contributions, making the software better for all.

However, open-sourcing previously proprietary software does have some technical considerations. For example, the software must be properly documented, the code must be clean and well-structured, and the software must be properly licensed. We saw from the interviews that these technical considerations are usually a deal-breaker and the reason why organisations do not open-source their software.

4.3.3 Hiring

To reduce the risk of using OSS, organisations should hire internal staff to manage OSS. Develop in-house proficiency and aim to reduce reliance on external

service providers by cultivating organisational expertise to oversee projects. This approach will enable quicker deployment of upgrades and enhancements (Team Srijan, 2010).

Smaller organisations might not necessarily hire new personnel but rather delegate the responsibility to an existing staff member who is recognised as a subject-matter expert within the specific area (Interviewee 8, 2024). This approach is viable because, as noted, "In general, open source developers are experienced users of the software they write. They are intimately familiar with the features they need, and what the correct and desirable behavior is" (Mockus et al., 2000). This innate understanding significantly mitigates one of the primary challenges in large software projects: the lack of domain knowledge. Consequently, smaller organisations can effectively rely on their in-house experts, capitalising on their comprehensive knowledge and experience.

Hiring is a significant investment, and it is not always feasible for smaller organisations to hire new personnel to manage OSS.

4.3.4 Culture

Open source has always been deeply rooted in culture, originating as a grassroots movement advocating for software freedom. Culture encompasses a broad spectrum, and individuals and organisations get involved in open source for various reasons.

When inquiring about the cultural drivers behind companies' engagement in open source, innovation emerged as the leading motivation, with 84% of respondents to Fintech Open Source Foundation State of Open Source Survey affirming it as a critical factor (Ellison et al., 2021).

Uber instituted internal standards for governing and incentivising contributing upstream and back to the community to encourage ongoing, regular involvement in open-source projects. Contributing back to the open-source community is one of the best ways to help ensure open-source project sustainability (*Uber Case Study* n.d.).

A big part of culture is fostering an environment where developers feel comfortable taking an unconventional route to solve a problem (*Autodesk Case Study* n.d.), and where they are encouraged to share code and collaborate with others (Abernathy, n.d.).

An organisation's culture is a significant factor in the adoption of open-source software, one that is quite hard to quantify and measure. There was very little mention of culture in the best practices we reviewed, and from our interviews, we found that there isn't an "open-source" culture but rather a more collaborative and innovative culture that is conducive to using open-source software.

4.3.5 Contributions

Active engagement in the open-source community offers substantial rewards. When you contribute to the projects your organisation utilises, you're boosting its reputation and playing a significant role in steering the software's development path. Such proactive participation can lead to the development of features that are beneficial to your organisation, as well as the resolution of issues that impact your operations.

Contributing to the open-source ecosystem extends beyond just coding. Offering documentation, identifying bugs, and aiding in translations represent other crucial ways to contribute. Through these contributions, you help cultivate a mutually beneficial relationship with the community (Yborra, 2024).

Contributing also helps keep the open-source component active and maintained. In recent years, a notable challenge has been the shortage of contributions or ongoing maintenance for projects, affecting even highly utilised packages like gorilla/mux, which have struggled to secure maintainers and consequently, had to archive their projects (Norblin, 2023).

Another way to contribute is through financial support. A recent survey by DigitalOcean found that only 20% of respondents had been paid for their contributions to open source, while 53% agreed or strongly agreed that individuals should be compensated for their work (DigitalOcean, 2022). This suggests that there is room for improvement in financial support for open-source contributors (Tandochain, 2023).

More work is being done to support financial contributions to open source; notable examples include GitHub Sponsors and Open Source Collective, both offering platforms for financial support of open-source projects (*Open Source Collective* 2024; *Explore GitHub Sponsors* 2024).

Contributing to the open-source community is a best practice for managing the security risks associated with OSS. Most of the literature we reviewed mentioned

contributing to the open-source community as a best practice. However, from our interviews, very few organisations contribute to the open-source community, and those that do do so in a limited capacity.

4.3.6 Hosting Events

Hosting events is a great way to engage with the open-source community. Events can be an excellent way to engage with the open-source community, and they can take many forms, from hackathons to conferences to meet-ups.

Schumacher, 2022, went as far as to say that corporate organisations looking to lead in open source should host events, as they are a great way to engage with the community and build relationships.

However, our interviews found that hosting events is not widely used in practice and that the best practices landscape does not reflect the real-world use of hosting events. Companies do send their employees to events, but larger organisations typically host events, as smaller organisations do not have the resources to host events.

5 Interviews

The research conducted for this paper included interviews with 8 individuals from a range of organisations. These included public sector, micro-enterprises and large corporations. The intention was to gain operational insight from users of OSS, the interviews focused on individuals with technical backgrounds and current user experience. We selected interviewees with technical backgrounds, and questions focused on their current user experience of dealing with open source software in an organisational context where open source is being used to develop software.

Participants included lead developers, chief technical officers, heads of development, directors, and project managers.

The interviews were semi-structured, with a set of questions that were asked to all interviewees, but with the flexibility to ask follow-up questions based on the responses given. You can see the full topic guide in Section 3.5.3.

The interviews were conducted over video calls and were recorded with the interviewees' consent. The interviews were then transcribed and analysed to identify common themes and patterns in the responses.

The interviews were conducted in the following order:

1. Interviewee 1: Lead developer at an educational platform
2. Interviewee 2: CTO at a micro-enterprise in the sustainable travel industry
3. Interviewee 3: Director at a web development agency
4. Interviewee 4: Product manager at a public sector organisation
5. Interviewee 5: CTO at a large educational platform
6. Interviewee 6: DevOps engineer at a large accounting firm
7. Interviewee 7: Head of development at a medium-sized IT support company
8. Interviewee 8: Director at a large web development agency

5.1 Findings

The interviews provided a wealth of information on the operational use of OSS and the risks associated with it. Through the information gained from these interviews, we were able to build an important picture of how a group of individuals perceive, use and manage open source software in different sectors

The interviewees unanimously emphasised the advantages of using OSS, highlighting the benefits of cost savings, flexibility, and innovation that it brought to their organisations.

It was noted that only two of the eight interviewed explicitly mentioned that they were actively looking for best practices online, or were following best practices. The rest of the interviewees mentioned that their organisational practices mostly stemmed from previous experience of the team or individual developers.

For consistency with Section 4, we have grouped the findings into the same categories: adoption, management, and community.

5.1.1 Adoption

Almost all of our participants described having an organisational process. However, there was no consistency in the process described, with some organisations interviewed having a formal process in place, whilst others relied on the discretion of the developers.

For example, Interviewee 1 mentioned that their organisation has a formal process for adopting OSS if the dependency is considered "business critical". If so, then the team will create a business case for that OSS component, which will then be reviewed by InfoSec, the principal engineer, and potentially the CTO. If the dependency is not considered "business critical", the team can adopt it without going through the formal process. However, the dependency will still be reviewed by the team in question, normally in a pull request (a request to merge the code into the main codebase). There was no formal process for gauging whether the dependency was safe to use, or how trustworthy it was. Trustworthiness was based on the team's previous experience with OSS components, and having a rough idea of desirable characteristics in an OSS component.

Interviewee 2, however, would only adopt an OSS component after building a minimal viable product (a basic version of the product with the minimum features required to be usable (MVP)) with it. If the MVP was successful, they would adopt it. To decide what OSS component to build the MVP with, they would look at the component's performance, stability, and operational cost. This was done ad-hoc and there was no formal process for this.

Interviewee 5 had no formal policy in place for adopting OSS, but they expressed their desire to have one.

Interviewee 6 had the most formal process for adopting OSS. First, the team would have to create an integrated design document with the integration pseudocode, security audit, and the reason for the adoption. This document then goes into an internal technical discussion with all stakeholders involved. Lastly, it goes to a security discussion with all the relevant stakeholders. From here, it is either approved or rejected. Interviewee 6 mentioned they understood why such a process existed, for security reasons, but they felt it was too cumbersome.

Interviewee 7 had no formal or structured adoption process in place. OSS adoption was driven primarily by a business need. If the team needed a new feature, they would look for an OSS component that could provide that feature, test the component, and if it were successful, they would adopt it.

Interviewee 8 had moved all responsibility for OSS adoption to the development teams, with no formal organisational process in place. The developers stated their main consideration was the connectivity and integration of the OSS component with the existing systems.

5.1.2 Management

When it came to the management of OSS, the interviewees had a range of practices in place. All participants used tooling to help manage the OSS components. However, there was little formal management practice in place.

Interviewee 1, Interviewee 2, Interviewee 3, Interviewee 4, Interviewee 6, and Interviewee 8 all used the same tool that integrated with their version control system to automatically updates dependencies in a repository and creates a pull request for the update.

Interviewee 1 and Interviewee 2 had no formal management approach in place outside of pull requests created by the tooling.

Interviewee 3's management process was mostly concerned with licensing and ensuring that any OSS component used was used in accordance with the license. They also mentioned that they had forked an OSS component, making a local copy of the component and modifying it to suit their needs, to make changes to it in the past. They have since moved away from forking due to the complexity of managing the forked component. As a general rule of thumb, they avoid small OSS components, instead opting to code the functionality themselves. Their main OSS dependency is PHP, a programming language used to build web applications, where they actively try to follow the best practices of the PHP community. As of March 2024, PHP has a total market share of 76.4% of all websites whose server-side programming language is known (W3Techs, 2024).

Interviewee 5 ran yearly pen tests on their products, all of which included OSS dependencies. They actively used tooling to help manage the OSS components, including tooling for automatically updating dependencies in a repository. Security checks were conducted during the pull request process, but their approach to security was mostly reactive, fixing issues as they arose.

Interviewee 6 mentioned that all risk was managed and mitigated by the team. There was no formal process in place to manage the risk of OSS components. Licensing was also down to the team to manage, with the organisation offering no support. They tried to follow the best practices outlined by their cloud vendor.

Interviewee 7 had no formal process in place for managing OSS components. They have run a forked version of an OSS component in the past but eventually adopted an alternative. They don't actively teach their developers about licensing

risks. They lean on tooling to help manage OSS components, using tooling for scanning, monitoring, and deployment.

Interviewee 8 relies mostly on tooling and external pen testers to manage the risk of OSS components. Any issues or vulnerabilities found were dealt with on a case-by-case basis, normally via a postmortem with the whole team. Licensing was addressed by one individual who was considered the expert in the organisation, and would be involved if a new OSS dependency was being added to any code base.

5.1.3 Community

Interviewee 1, Interviewee 5, Interviewee 6, and Interviewee 7 all mentioned that they or their organisation do not contribute to the OSS community. They all felt obligated to contribute back to the OSS community but did not have the time or resources to do so.

Interviewee 2 only contributed to the OSS community if a bug or a missing feature blocked them. They didn't offer any financial support to the OSS community.

Interviewee 3 occasionally contributed to the OSS community, but it was not their priority. They didn't offer any financial support to the OSS community. However, they do attend conferences to stay up-to-date with the latest developments in the OSS community.

Interviewee 4 stated that community engagement was an internal policy and actively promoted. However, they "struggled" to get developers to engage with the community. There was an element of risk in open-sourcing code, as it could expose the organisation to criticism or reveal sensitive code.

Interviewee 8 had the most proactive approach to community engagement. They actively fostered a culture of contributing back to the OSS community, with certain members of staff considered "experts" for that OSS component. These experts were the ones who would contribute back to the OSS community but also acted as the knowledge base in the organisation for that OSS component. They were active in the community, fixing bugs, contributing to discussions, and helping other users of the OSS. The organisation even sent the experts to conferences to represent the organisation and give talks. Although Interviewee 8 didn't explicitly mention the benefits of such an extensive community engagement, they did mention they had

"some of the best developers" in their city.

5.2 Remarks

The interviews provided a wealth of information on the use of OSS and the risks associated with it. Significantly, there was no consistency in the adoption and management of OSS, with each organisation having its own approach. However, we did see the same tools being used and mentioned, to help with vulnerability scanning and dependency management.

There was a general lack of community engagement, with most participants saying that their organisations did not contribute to the OSS community. This was mostly due to a lack of time and resources and a perceived risk in open-sourcing code.

We also found that outside of Interviewees 3 and 6, nobody was actively looking for best practices online. Typically, organisational practices mostly stemmed from previous experience of the team/organisation.

6 Findings

6.1 Broad and Evolving Guidance

Overall, the guidance on open-source software (OSS) management is broad, with most sources recommending similar best practices for adopting, managing, and engaging with the community. These aspects have been covered in depth in literature, with a wide range of best practices and tools available to help organisations manage the security and compliance risks associated with OSS.

However, the guidance is not static, and it is continually evolving. For example, introducing the executive order requiring SBOMs in all federal software (The White House, n.d.) has led to a surge in the popularity of Software Bill of Materials (SBOMs) as well as the amount of guidance and best practice recommendations on how to use them. Something that was not as widely recommended or used previously.

6.2 Lack of Industry-Specific Best Practices

We found, although broad, that OSS best practices seem lacking when considering the context in which OSS components are used, with most guidance being generic and not tailored to specific sectors.

A good example of this is the difference between the finance and education sectors.

The finance industry has seen a surge in OSS adoption, driven by the need for innovation, cost reduction, and regulatory compliance. The Fintech Open Source Foundation (FINOS) is pivotal in fostering OSS adoption within this sector. It provides a platform for collaboration among competing financial institutions, enabling them to work on mutual projects while adhering to regulatory standards (Griswold, 2023).

However, outside of finance, there is a lack of guidance on managing OSS components in specific industries, such as education. Perhaps this is due to the fact that the finance industry is more heavily regulated than other industries, and therefore, there is a more substantial need for more guidance.

A study into adopting OSS in the education sector found that adopting OSS allows for its customisation to meet local needs. For instance, Open Office in India has been adapted to include support for twenty-six Indian languages, surpassing the twelve languages supported by Microsoft Office (Kumar, 2007). Mainly, OSS can be used in education to enable students to share, participate, modify, and contribute to software development tailored to the needs of the local market and community (Makhathe et al., 2013).

Gupta et al., 2018, found that only 20% of teachers surveyed in Delhi were familiar with the different types of software licenses, and 45% of the teachers thought that OSS could be beneficial to school education. However, the most significant barriers to adopting OSS in schools are lack of knowledge, training, availability of technology, and awareness of such software (Gupta et al., 2018).

One could argue that establishing a centralised open-source body or charity within the education sector to share knowledge and understanding could help address these issues.

We can see how collaboration across industries and sectors can be beneficial. In the public domain, the US Cybersecurity and Infrastructure Security Agency

(CISA) has initiated working groups comprising multiple stakeholders focused on software security. These groups bring together specialists from different industries to jointly create tools for enhancing software security, such as guidelines and frameworks for SBOMs and Vulnerability Exploitability Exchange (VEX) standards. Through collaborative endeavours with these working groups and through community gatherings, US CISA has facilitated the creation of resources for SBOM and VEX, which have gained widespread adoption across both public and private sectors (Osborne et al., 2023). Although not specific to one industry, the work done by the CISA is an excellent example of how industry-specific and scale-appropriate best practices can be developed and implemented, through multi-stakeholder engagement.

6.3 Lack of Scale-Appropriate Best Practices

We also found a disconnect between the best practices and the real-world use of OSS components, with the best practices not always being reflective of the real-world use of OSS as described by participants in our interviews. Smaller organisations struggle to meet the best practices outlined in existing guidance due to limited financial and workforce resources.

Since around 1999, large tech companies, such as IBM, began to embrace and support the concept of open-source software (Codemotion, 2023). Inevitably, due to these companies' financial backing and resources, they have been able to exert a significant influence on the open-source community. The influence of large tech companies on the open-source community could also result in these corporations influencing the direction of open-source projects to align with their business interests rather than the community's interests.

One such example is Microsoft "releasing a competing open source IDE (Visual Studio Code) of their own has enabled them to set the standard for IDE platforms, attract new users that will be exposed to Microsoft development platforms that can be easily integrated to other Microsoft platform later on," [sic] (Habusha, 2019).

It is nearly impossible for hobbyist programmers and smaller special interest groups (SIGs) to compete with these giant corporations, who have the money needed to develop and maintain open-source projects at a higher level (Bridgwater, 2019).

This dominance poses risks, particularly in setting high standards and best prac-

tices that smaller businesses struggle to meet due to limited financial and workforce resources. Moreover, by introducing their own tools, such as Microsoft with Visual Studio Code (Habusha, 2019), these corporations not only set new industry benchmarks but also steer the development community towards their ecosystems. This trend towards standardisation and potential monopolisation can sideline SMEs and hobbyist contributors, creating an environment where innovation and diversity may be compromised in favour of commercial interests and the dominance of a few large players.

We saw this first-hand from the disconnect between what the literature suggested were best practices and what we found from our interviews. This lack of specificity was observed across small startups with merely single-figure employees to larger entities with significant eight-figure valuations. There existed a discernible absence of formalised processes or best practices.

Certain best practices, like regular vulnerability assessments and static application security testing, are resource-intensive and require a significant investment in personnel unless automation is used. Most SMEs do not have the capacity to have an internal OSS team or an open-source program office (OSPO).

It's important to clarify that this doesn't imply a complete absence of process or best practice. However, the identified processes tended to originate from the developers' or teams' internal practices, opinions or experiences within the organisation, rather than being drawn from academic research or well-established best practices.

We believe that the best practices landscape does not reflect the real-world use of OSS, and more work should be done to consider the different approaches to risk management and mitigation based on the size of the organisation.

We can see that outside highly regulated industries, there is a lack of guidance on how to manage OSS components in specific industries, such as education. As the guidance is broad, it could imply that more sector-specific guidance is not required. However, the disconnect between the best practices and the real-world use of OSS components, with the best practices not always being reflective of the real-world use of OSS, is problematic.

6.4 Disagreements and Diversity in Approaches

Adopting, managing, and contributing to open-source software is a complex and multifaceted process, and there are many different approaches to managing the security and compliance risks associated with OSS.

As "open-source software" is such a broad term, ranging from small hobbyist projects to large corporate-backed projects, it is unsurprising that there are many different approaches to managing the security and compliance risks associated with OSS.

For example, the risk and mitigation approach varied from organisation to organisation. Larger organisations seemed more proactive in mitigating risk, with dedicated teams and processes in place. Whereas smaller organisations were more reactive, only taking action when a risk materialised and looking to improve their processes after the event.

We've seen how there are multiple different maturity models one can use to assess the maturity of an organisation's open-source software management. Further work is suggested to create a standardised maturity model for open-source software management.

We've also seen how the best practices do not always reflect the real-world use of OSS. This could be due to the influence of large tech companies on the open-source community and the dominance of these companies in setting standards and best practices that smaller businesses struggle to meet, due to limited financial and workforce resources. It could also be due to the lack of organisational guidance, thus leaving it to the developer or team to decide on the best practices.

Critically, the lack of a formal process for judging an open-source trustworthiness is an oversight. This is particularly important given the increasing reliance on open-source software and the increasing number of vulnerabilities found in open-source software.

Open-source being such a broad term, it is unsurprising that there are many different approaches to managing the security and compliance risks associated with OSS. However, we find certain areas around the maturity model, the influence of large tech companies, and the lack of a formal process for judging open-source trustworthiness to be an oversight in the best practices literature.

7 Recommended Best Practices

We have conducted a comprehensive review of the literature and interviewed experts to identify the best practices for managing open-source software (OSS) components. Our main takeaways are threefold:

- Broad yet Evolving Guidance (Section 6.1)
- Lack of Industry-Specific Best Practices (Section 6.2)
- Lack of Scale-Appropriate Best Practices (Section 6.3)
- Disagreements and Diversity in Approaches (Section 6.4)

Building upon these findings, we evaluated the current literature by scoring each best practice in accordance with our scoring methodology. The scoring methodology assigned a score of 0 (non-existent), 0.33 (basic), 0.66 (intermediate), or 1 (comprehensive) to each aspect of OSS; adoption, management, community engagement, and context.

This scoring methodology allowed us to identify the most important aspects of OSS management and community engagement. We found the most important aspects of OSS are:

- Establishing what metrics to use to evaluate the trustworthiness and maturity of OSS components.
- Understanding the software supply chain, including the dependencies a software product uses.
- Continuously monitoring the software supply chain, checking for vulnerabilities, licensing issues, and new versions of OSS components.
- Engaging with the open-source community through financial contributions, documentation, identifying bugs, or aiding in translations.

To achieve these key points we've identified four essential best practices for managing OSS components: establishing an internal OSS policy, creating a Software Bill of Materials (SBOM), continuous monitoring and reviewing, and community engagement.

7.1 Open Source Software Policy

Most companies interviewed lacked a consistent and formalised process for OSS adoption, relying instead on informal practices or developer discretion. Of the companies interviewed, all also lacked a formal standard to evaluate the maturity and trustworthiness of OSS components. The lack of a formal process can be risky, as it may lead to adopting components with security vulnerabilities or licensing issues (Schofield, 2008; Gatto, 2019).

The literature review found that developers are mainly concerned with the quality of the open-source component (Hou et al., 2022). However, from our interviews, there was no consistent way to evaluate the quality of an OSS component. The quality gauge of an OSS component/dependency should be derived from a formalised internal policy and not reliant on an individual developer's discretion. This shift would transfer the decision-making process from the developer to the organisation, shifting the burden of responsibility from the developer to the organisation and ensuring that any decision is safe and in the organisation's best interest.

69% of organisations surveyed by the Linux Foundation (Hendrick et al., 2023) who have some kind of OSS policy or initiative in place found that is was effective in managing the risk of OSS components, and it significantly improved OSS security. Organisations that adopt an OSS policy see across the board improvements in OSS security, license compliance, and risk management (Hendrick et al., 2023).

We recommend that all organisations establish an internal OSS policy that details the criteria for evaluating the trustworthiness and maturity of OSS components.

Care should be taken to ensure that the policy is not overly prescriptive, as this may stifle innovation and discourage the use of OSS. Furthermore, the approach should be nuanced, as different projects may require different levels of scrutiny. Balancing the need for security and compliance with the need for innovation and agility is crucial to the policy's success. The decision-making process needs a clear and transparent approval step, as outlined in Section 4.1.4. The policy should be a living document, subject to regular review and updates.

From this, we've highlighted five points that were frequently recommended in the literature, or aspects of the interviews that were lacking in the organisations we

spoke to. The specific needs for the OSS policy may change from organisation to organisation, but the following elements should be considered:

- **A list of acceptable licenses:** The policy should specify which licenses are acceptable for use for both internal and external projects.
- **An approved open source list:** The policy should state whether OSS can be used in what proprietary software/external projects.
- **Criteria for evaluating OSS components:** The policy should detail the criteria for assessing the trustworthiness and maturity of OSS components.
- **Severity levels:** As noted by Larios Vargas et al., 2020, "the first factor taken into account is the type of project.". The policy should guide the process for different risk levels of OSS components. For example, a simple OSS dependency might forego a formal approval process, while a critical OSS component might require a more thorough evaluation.
- **An approval process:** The policy should outline the approval process for using OSS components, which stakeholders are involved in, and the criteria for approval. As mentioned by Butler et al., 2022, the approval process relies more on assessment and discretion than on strict, predefined criteria.

Henceforth, with an internal OSS policy, organisations can expect to have a formalised process for adopting OSS components that is consistent across the organisation. Organisations will also be able to set constraints on using specific licenses that might not be suitable for the organisation and will ensure that judging the trustworthiness and maturity of OSS components is not left to developers' discretion. Most importantly, the policy creation should include stakeholders and developers from the start, ensuring it is tailored to the organisation.

7.2 Software Bill of Materials

Another issue with managing OSS components is the need for more visibility into the software supply chain. Without a clear understanding of the software supply chain, organisations are at risk of "dependency hell"; the bigger the system, the more it grows, the more dependencies you integrate into the software (Mishra, 2023). The lack of visibility into the software supply chain can lead incidents like the "left-pad" incident; the withdrawal of a minor package from the npm package manager resulted in widespread failure in numerous projects (Williams,

2016). Therefore, the better the understanding of the supply chain, the better the organisation can manage the risks associated with OSS components.

A Software Bill of Materials (SBOM) is a list of OSS components used in a software product, including their dependencies and associated licenses. An SBOM provides visibility into the software supply chain, enabling organisations to identify and mitigate potential security and licensing issues. The benefits of SBOMs include lower costs, reduced code bloat (unnecessarily long, slow, or otherwise wasteful of resources), and improved security (United States Department of Commerce, 2021).), and ensuring a more secure software supply chain (Reflectiz, 2022). It also ensures there is a source of truth for the components used in your software, which can be used to enforce the internal OSS policy.

We appreciate that creating an SBOM can be a time-consuming and resource-intensive process. We suggest organisations look to integrate an SBOM into their development and deployment processes, that is generated automatically, to alleviate the resource burden. However, there are open-source and free tools available that can help organisations to build SBOM generation into their development processes in a more cost-effective way.

The United States Department of Commerce, 2021 has released a guidance document on creating an SBOM, which includes best practices for creating an SBOM. The two primary formats for a SBOM are SPDX and CycloneDX. Both formats are popular, yet organisations are advised to select the format that most aptly meets their specific requirements (Alvarenga, 2023a; Springett, n.d.).

Visibility into the software supply chain is key for managing the risks associated with OSS components. To achieve this, organisations should consider generating a SBOM for their software products. This will provide a clear understanding of the software supply chain and help in enforcing an internal OSS policy. We are, of course, cognisant that this process can be time-consuming and costly, organisations should look to automate the process of generating an SBOM and look to a tool to help with this process.

7.3 Continuous Monitoring

Modern software is a complex web of dependencies, with updates and patches released regularly. It is important to continuously monitor the software supply chain for vulnerabilities, licensing issues, and new versions of OSS components. Without continuous monitoring, organisations are at risk of data breaches or se-

curity incidents, seen in the equifax data breach (Fruhlinger, 2020) and the Log4j vulnerability (Gallo, 2022).

To minimise risk, organisations should continuously monitor their software supply chain for vulnerabilities, licensing issues, and new versions of OSS components.

Determining what OSS component is being used within a software application, to then monitor them continuously, is a complex and time-consuming endeavour. This is the area where an organisation can get the most benefit from an software composition analysis (SCA) tool (Ombredanne, 2020), which will automate this process, saving time and resources.

Continuous monitoring can be achieved by using a SCA tool. SCAs can detect important issues like security vulnerabilities, licensing problems or outdated library versions (Molin et al., 2023). This process involves scrutinising the entire software supply chain, which includes proprietary code and third-party components like open-source libraries, to address several principal concerns.

An SCA tool's effectiveness is underpinned by an SBOM, therefore, creating an accurate SBOM is essential. An accurate SBOM aids in vulnerability detection and compliance verification (Alvarenga, 2023b). Hence why we recommend that organisations that work with OSS adopt an SBOM.

When deciding on an SCA tool, organisations should take care to ensure they integrate with any existing security infrastructure and prioritise features like accuracy and scalability (Alvarenga, 2023a).

Continuous monitoring, in the form of a SCA tool, helps enforce an organisational OSS policy. 52% of organisations surveyed by the Linux Foundation in 2023 (Hendrick et al., 2023), who had a formal OSS policy, used automation for monitoring license compliance. 58% of those surveyed used automation for monitoring security vulnerabilities, showing that continuous monitoring works in tandem with an OSS policy.

Due to the complex nature of modern software, continuous monitoring is key to ensure that an organisation stays up-to-date with the latest vulnerabilities, licensing issues, and new versions of OSS components. As a manual process, continuous monitoring can be time-consuming and resource-intensive. However, by adopting a SCA tool, organisations can automate the process of continuous monitoring, saving time and resources. An SCA tool works in tandem with both an

internal OSS policy and an SBOM.

7.4 Tooling

By adopting the above best practices, organisations will be well-placed to manage their OSS dependencies effectively. However, we know that adopting these best practices would require a resource-intensive effort for small and medium-sized enterprises.

Most interviewees, however aware of the risk of OSS, needed more resources to manage the risk, or need to take the proper steps to manage the risk. Tooling is a great leveller between large and smaller enterprises, as it can automate the process of managing OSS components. Through the use of tooling, organisations can greatly reduce the time and resource burden of introducing OSS best practices.

We recommend leveraging tooling as a means to achieve the list above (OSS Policy, SBOM, and Continuous Monitoring). Tooling can be used to automate the process of managing OSS components, can be used to enforce an internal OSS policy, perform automated and continuous monitors, and generate and maintain an SBOM. This will significantly reduce the burden on developers and ensure that the SBOM is always up-to-date and the organisation complies with the internal OSS policy.

Each tool can be utilised to achieve enforcement of an internal OSS policy, generate and maintain an SBOM, and perform continuous monitoring. The most effective tools currently available on the market have the following features (Worthington et al., 2023):

- Secure libraries and the software supply chain, focussing on preventing supply chain attacks.
- Offer solutions for source code scanning, including both on-premises and hosted options.
- Include workflows tailored for development, security, and compliance teams.
- Feature advanced capabilities for assessing third-party code, either through binary analysis or specialised add-ons.
- Plan to enhance capabilities related to Software Bill of Materials (SBOM) and vulnerability detection.

- Utilise comprehensive policy engines for detailed risk management and company policy enforcement.
- Aim to integrate security seamlessly into development workflows, minimising disruption.
- Focus on remediation, offering automation and educational resources to address vulnerabilities.
- Designed to support diverse software supply chains, regulated industries, and environments prioritising DevSecOps and frequent deployments.

Although tooling could potentially introduce a cost implication through a monthly subscription, some of the tools have a free tier. Any cost-benefit analysis by organisations looking to use these tools to manage OSS usage should consider the cost of manually implementing the best practices listed and the potential cost of a security incident or cyberattack.

Using one of the above tools will significantly reduce the burden on developers while ensuring that best practices are followed.

7.5 Community Engagement

Community engagement is an organisational best practice when it comes to OSS, that is often overlooked. As mentioned in Section 4.3.5, it can extend beyond just coding. Offering documentation, identifying bugs, and aiding in translations represent other crucial ways to contribute (Yborra, 2024), as well as financial contributions.

Unfortunately, our interviews have uncovered the majority of those interviewed do not actively give back to the OSS community.

We recommend that organisations actively engage with the OSS community, as this can provide a range of benefits.

Firstly, community engagement can help organisations attract new talent and lower costs for hiring (Nagle, 2018). It can help level the playing field for smaller organisations, who may not have the resources to compete with larger organisations in terms of hiring and developing software (Wright et al., 2023). It can help boost entrepreneurship, and innovation, within an organisation (Wright et al., 2023; Haefliger et al., 2008; Hendrick et al., 2023). It can even enhance an organisation's credibility and reputation, positioning it as a knowledgeable and up-to-date leader

in software development (Varian et al., 2003). Community engagement serves to not only increase the quality of the OSS components that the organisation uses, but also ensures a lasting and sustainable OSS community (Varian et al., 2003).

Fostering a culture of community engagement can be achieved through various activities, including contributing to OSS projects, participating in community events, or providing financial support to OSS projects. From this engagement, organisations can raise the quality of their internal developers and attract new talent. Not to mention increasing the quality of the OSS components that the organisation uses.

Although beneficial, we appreciate that giving back to the open-source community is not always feasible for all organisations. This is primarily down to the resources and time required to contribute to the community. Unfortunately, there is no clear solution to this, and we recommend that organisations contribute to the OSS community to the best of their ability. However, a few initiatives can be taken to foster a culture of community engagement.

Open Source Friday is a community engagement initiative by GitHub. Open Source Friday encourages organisations to dedicate developer time each Friday to contribute to OSS projects and is an excellent example of a community engagement initiative (GitHub, n.d.).

Another initiative is the GitHub Sponsors program, which allows organisations to financially support OSS projects (*Explore GitHub Sponsors* 2024).

We believe that community engagement is key to raising the quality of organisations' OSS components. This is an under-sold opportunity for organisations to simultaneously improve the quality of the OSS components they use, increase their credibility, boost innovation, and lower the costs of hiring new employees. It can also level the playing field away from the big technology companies dominating the space. With increased community engagement from all who use OSS, the OSS community can be more diverse and inclusive.

It is our belief that there exists an opportunity for government to offer guidance on the benefits of community engagement and the different ways organisations can contribute to the OSS community. Within the group of people we interviewed there was a recognition of the need to contribute. However, there was a lack of understanding regarding the benefits this brings to their own organisations. Therefore, we suggest that further efforts are needed to increase awareness about the advantages of community engagement and the various forms of participation

available to organisations within the OSS community.

8 Conclusion

This report has examined the practices surrounding the adoption, management, and community engagement of open-source software (OSS), drawing on a broad literature review and insights from industry practitioners.

We found open-source best practices are broad yet evolving. We highlighted two significant gaps in the landscape: the lack of industry-specific and scale-appropriate best practices, and the disagreements and differences in approaches.

We found guidance on OSS adoption and management being introduced, promoted, and aimed at big technology companies. Smaller organisations interviewed were often either not following best practices or are following their own internal practices, not aligned with the best practices outlined by the wider community.

Although best practices are broad, we found there was a lack of consensus on the best approach to managing OSS components. We found this mostly due to the lack of a one-size-fits-all approach, with different organisations having different needs and requirements.

We have produced a list of the most important of these best practices whilst still trying to factor in the context of different organisational sizes. We have recommended that organisations establish an internal OSS policy, create a Software Bill of Materials, continuously monitor their software supply chain, and promote engagement with the OSS community.

In recognition of the potential increased time and resource pressures, especially for smaller organisations, we also recommend leveraging tooling to automate the process of managing OSS components. This will significantly reduce the burden on developers, ensure the organisation complies with the internal OSS policy, keeps the SBOM up-to-date, and continuously monitors for vulnerabilities and licensing issues.

We believe that the best practices we have recommended will reduce the risk of using OSS, improve the quality of OSS components, and enhance the security of the software supply chain. Whilst still being achievable for organisations of all sizes.

9 Future Work

Looking ahead, there are several areas where further research and development will enhance the management of open-source software (OSS) components:

1. Encouraging organisations to adopt the best practices for managing OSS components.
2. Developing best practice guidance suitable for a wide range of organisation sizes/scopes.
3. Promoting organisations on the benefit of contributing back to the OSS community
4. Exploring if industry-specific guidance is needed for managing OSS components.
5. Investigating the long-term impacts of community engagement on open-source quality and security from smaller organisations.
6. Exploring the development of a standardised open-source adoption metric list that can be used to measure the maturity and trustworthiness of OSS components.

We believe that these areas of research will help to improve the management of OSS components and the security of the software supply chain, whilst also benefiting organisations of all sizes and sectors that use OSS components.

References

OpenLogic (2023). *2023 State of Open Source Report*. Perforce Software, Inc. Retrieved from https://www.openlogic.com/resources/2023-state-open-source-report.

Schofield, Jack (Dec. 2008). *FSF sues Cisco over Linksys open source code*. https://www.theguardian.com/technology/blog/2008/dec/12/cisco-fsf-opensource. Accessed: 2023-04-03.

Osborne, Cailean et al. (Sept. 2023). *The European Public Sector Open Source Opportunity*. Tech. rep. Foreword by Gabriele Columbro, Linux Foundation Europe. Accessed: 4 March 2024. The Linux Foundation. URL: https://www.linuxfoundation.org/the-european-public-sector-open-source-opportunity.

Hammond, Jeffrey (Mar. 2009). *Open Source Adoption: What Your Peers Are Up To*. https://www.eclipse.org/org/foundation/membersminutes/20090326StrategySummit/OSS-WYPAUT3_JeffreyH.pdf. Presented at the Eclipse Foundation Strategy Summit, March 24-25, 2009.

Umm-e-Laila et al. (Dec. 2016). "Comparison of Open Source Maturity Models". In: *Proceedings of the 8th International Conferences on Advance in Information Technologies (IAIT 2016)*. University of Macau. Macau, China. URL: https://www.researchgate.net/publication/318339374_Comparison_of_Open_Source_Maturity_Models.

Hou, F. et al. (Nov. 2022). "A systematic literature review on trust in the software ecosystem". In: *Empirical Software Engineering* 28.8. Accessed from: Google Scholar. URL: https://www.researchgate.net/publication/365699098_A_systematic_literature_review_on_trust_in_the_software_ecosystem.

Spinellis, Diomidis (Nov. 2019). "How to Select Open Source Components". In: *IEEE Software*. URL: https://ieeexplore.ieee.org/document/8909944/authors.

Butler, Simon et al. (Sept. 2022). "Considerations and challenges for the adoption of open source components in software-intensive businesses". In: *Empirical Software Engineering* 28.8. URL: https://www.sciencedirect.com/science/article/pii/S0164121221002442#b37.

System Package Data Exchange (SPDX®) (2024). https://spdx.dev. Accessed: 2024-03-05. The Linux Foundation.

CycloneDX (2024). https://cyclonedx.org/. Accessed: 2024-03-05. OWASP Foundation.

United States Department of Commerce (July 2021). *The Minimum Elements For a Software Bill of Materials (SBOM)*. Pursuant to Executive Order 14028 on Improving the Nation's Cybersecurity. URL: https://www.ntia.gov/blog/ntia-releases-minimum-elements-software-bill-materials.

NTIA Multistakeholder Process on Software Component Transparency Framing Working Group (Oct. 2021). *Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) - Second Edition*. Working Group Report. National Telecommunications and Information Administration. URL: https://www.ntia.gov/files/ntia/publications/ntia_sbom_framing_2nd_edition_20211021.pdf.

Murray, Adam (Jan. 2022). "Open Source Licenses in 2022: Trends and Predictions". In: *Mend Blog*. Accessed: 2024-03-05. URL: https://www.mend.io/blog/open-source-licenses-in-2022-trends-and-predictions/.

Mathpati, Ninad (Feb. 2023). *Risks of Open-Source Software*. https://cobalt.io/blog/risks-of-open-source-software. Accessed: 2024-03-05.

The Linux Foundation (2024). *Tools for Managing Open Source Programs*. https://www.linuxfoundation.org/tools-for-managing-open-source-programs/. Accessed: 2024-03-05.

Sawers, Paul (Apr. 2021). "84% of codebases contain an open source vulnerability". In: *VentureBeat*. URL: https://venturebeat.com/2021/04/13/synopsys-84-of-codebases-contain-an-open-source-vulnerability/.

Hackerone (Jan. 2024). *What Is Vulnerability Assessment? Benefits, Tools, and Process*. https://www.hackerone.com/knowledge-center/what-vulnerability-assessment-benefits-tools-and-process. Accessed: 2024-03-05.

Ponta, Serena Elisa et al. (Sept. 2020). "Detection, assessment and mitigation of vulnerabilities in open source dependencies". In: *Empirical Software Engineering* 25.5. URL: https://dl.acm.org/doi/10.1007/s10664-020-09830-x.

Wen, Shao-Fang et al. (2019). "An Empirical Study of Security Culture in Open Source Software Communities". In: *2019 IEEE/ACM International Confer-*

ence. IEEE. URL: https://ieeexplore.ieee.org/document/9073010.

Department of Defense (Jan. 2022). *Software Development and Open Source Software*. Memorandum. Chief Information Officer Memorandum for Senior Pentagon Leadership, Commandant of the Coast Guard, Commanders of the Combatant Commands, Defense Agency and DoD Field Activity Directors. 6000 Defense Pentagon, Washington, D.C. 20301-6000. URL: https://dodcio.defense.gov/Portals/0/Documents/Library/SoftwareDev-OpenSource.pdf.

Daigle, Kyle (Nov. 2023). *Octoverse: The state of open source and rise of AI in 2023*. https://github.blog/2023-11-08-the-state-of-open-source-and-ai/. Accessed: 2024-03-05.

Wainstein, Limor (June 2018). "7 Best Practices for Managing Open Source Components". In: *Engineering*. Accessed: 2024-03-05.

Williams, Chris (Mar. 2016). *How one developer just broke Node, Babel and thousands of projects in 11 lines of JavaScript*. Accessed: 2024-03-05. URL: https://www.theregister.com/2016/03/23/npm_left_pad_chaos/.

OWASP (2020). *OWASP Top Ten*. https://owasp.org/www-project-top-ten/. Accessed: 2024-03-05.

Zorz, Zeljka (2020). *Number of open source vulnerabilities surged in 2019*. https://www.helpnetsecurity.com/2020/03/13/open-source-vulnerabilities-2019/. Accessed: 2024-03-05.

Contrast Security (Apr. 2020). *Understanding the Risks of Open-Source Software*. https://www.contrastsecurity.com/hubfs/Understanding-the-Risks_WhitePaper_042020_Final.pdf?hsLang=en. Accessed: 2024-03-05.

Chandler, Paul A. et al. (Oct. 2022). *The Importance of Tracking and Managing the Use of Open Source Software*. https://www.mayerbrown.com/en/insights/publications/2022/10/the-importance-of-tracking-and-managing-the-use-of-open-source-software. Accessed: 2024-03-05.

Anand, Megha (Sept. 2023). *Seven Best Practices to Secure Your Open Source Components*. https://security.salesforce.com/blog/seven-best-practices-to-secure-your-open-source-components. Accessed: 2024-03-05.

The Open Source Definition (2024). https://opensource.org/osd. Accessed: 2024-03-05.

Méndez-Tapia, Lucía et al. (2021). "Organizational Readiness Assessment for Open Source Software Adoption". In: *Proceedings of the International Conference on Knowledge Management and Information Systems*. Cuenca, Ecuador; València, Spain: SCITEPRESS – Science and Technology Publications. URL: https://www.scitepress.org/Papers/2021/104970/104970.pdf.

TODO Group (June 2023). *Open Source Program Office Definition*. https://ospoglossary.todogroup.org/ospo-definition/. Accessed: 2024-03-04.

Walker, John Mark (2016). *What is an open source program office? And why do you need one?* Accessed: 2024-03-04. URL: https://opensource.com/business/16/5/whats-open-source-program-office (visited on 03/04/2024).

Uber Case Study (n.d.). Accessed: 2024-03-04. URL: https://todogroup.org/resources/case-studies/uber/ (visited on 03/04/2024).

Hsieh, Brian (n.d.). *Uber and Open Source*. TODO Group Case Studies. Available at: https://todogroup.org/resources/case-studies/uber/.

Team Srijan (June 2010). *Migrating to Open Source - A Survival Guide for SMEs*. https://materialplus.srijan.net/resources/blog/migrating-open-source-survival-guide-smes.

Interviewee 8 (Feb. 2024). *Interview with Interviewee 8*.

Ellison, Tosha et al. (Oct. 2021). *The 2021 State of Open Source in Financial Services*. https://www.finos.org/reports/2021-state-of-open-source-in-financial-services. Fintech Open Source Foundation (FINOS).

Autodesk Case Study (n.d.). Accessed: 2024-03-04. URL: https://todogroup.org/resources/case-studies/autodesk/ (visited on 03/04/2024).

Abernathy, Christine (n.d.). *Uber and Open Source*. Meta Case Study. Available at: https://todogroup.org/resources/case-studies/meta/.

Mockus, Audris et al. (June 2000). "A case study of open source software development: the Apache server". In: *Proceedings of the 22nd international conference on Software engineering*. ACM. DOI: 10.1145/337180.337209. URL: https://dl.acm.org/doi/10.1145/337180.337209.

Landwerth, Immo (Nov. 2014). *.NET Core is Open Source*. https://devblogs.microsoft.com/dotnet/net-core-is-open-source/.

Foley, Mary Jo (Aug. 2016). *Microsoft open sources PowerShell; brings it to Linux and Mac OS X*. https://www.zdnet.com/article/microsoft-

open-sources-powershell-brings-it-to-linux-and-mac-os-x/. Senior Contributing Editor.
Lebaredian, Rev (Dec. 2018). *NVIDIA Extends PhysX for High-Fidelity Simulations, Goes Open Source*. https://blogs.nvidia.com/blog/physx-high-fidelity-open-source/.
ARS STAFF (Dec. 2015). *Microsoft Open-Sources Live Writer, Beloved but Abandoned Blogging Tool*. Accessed: 2024-03-07. URL: https://arstechnica.com/information-technology/2015/12/microsoft-open-sources-live-writer-beloved-but-abandoned-blogging-tool/.
Yborra, Marine (July 2024). *Managing open source software integration in software development*. URL: https://vaultinum.com/blog/managing-open-source-software-integration-in-software-development#best-practices-for-managing-open-source-software-integration.
Norblin, Jonathan (May 2023). *Giving back to Free and Open-Source Software (FOSS)*. URL: https://www.scaleway.com/en/blog/foss-giving-back/ (visited on 05/16/2023).
Open Source Collective (2024). https://www.oscollective.org/. Accessed: 2024-03-05.
Explore GitHub Sponsors (2024). https://github.com/sponsors/explore. Accessed: 2024-03-05.
DigitalOcean (2022). *The 2022 Report on Open Source and Developer Trends*. https://anchor.digitalocean.com/rs/113-DTN-266/images/DigitalOcean-Currents_June-2022.pdf. Accessed: 2024-03-05.
Tandochain (May 2023). "The Importance of Financial Support for Open-Source Software". In: Accessed: 2024-03-05.
Schumacher, Cornelius (2022). *The five stages of corporate open source adoption*. https://www.youtube.com/watch?v=6UMCe5o0KBw. Accessed: 2024-03-05.
Griswold, Grizz (Nov. 2023). *FINOS Annual Report Training Certification*. https://www.finos.org/press/finos-annual-report-training-certification. Accessed: 2024-03-05. FINOS/The Linux Foundation.
Gupta, Deepty et al. (2018). "ADOPTING FREE AND OPEN SOURCE SOFTWARE (FOSS) IN EDUCATION". In: *i-manager's Journal of Educational Technology* 14.4, p. 53. URL: https://www.researchgate.net/

publication/324893991_ADOPTING_FREE_AND_OPEN_SOURCE_SOFTWARE_FOSS_IN_EDUCATION.

Kumar, V. Sasi (Oct. 2007). *Free Software in Kerala*. URL: https://egovindia.wordpress.com/category/foss/.

Makhathe, Mohlamme et al. (Jan. 2013). "ICT Students' Perception Concerning Free and Open Source Software: A Case Study of Central University of Technology". In: *the 15th International Conference on Advanced Communication Technology (ICACT)*. Central University of Technology. URL: https://www.researchgate.net/publication/235605928_ICT_Students'_Perception_Concerning_Free_and_Open_Source_Software_A_Case_Study_of_Central_University_of_Technology.

Codemotion (Feb. 2023). *Infographic: The History of Open Source*. Codemotion Magazine. Infographics. URL: https://www.codemotion.com/magazine/infographics/the-history-of-open-source/.

Ombredanne, Philippe (Oct. 2020). *Free and Open Source Software License Compliance: Tools for Software Composition Analysis*. URL: https://ieeexplore.ieee.org/ielx7/2/9206399/09206429.pdf.

Dietrich, Jens et al. (June 2023). *On the Security Blind Spots of Software Composition Analysis Caused by Cloning and Shading*. https://arxiv.org/pdf/2306.05534.pdf. Accessed: 2023-09-30.

Fruhlinger, Josh (Feb. 2020). *Equifax data breach FAQ: What happened, who was affected, what was the impact?* https://www.csoonline.com/article/567833/equifax-data-breach-faq-what-happened-who-was-affected-what-was-the-impact.html.

Molin, Andy et al. (Oct. 2023). "Assessing the Real Impact of Open-Source Components in Software Systems". In: *IEEE Access*.

Wright, Nataliya Langburd et al. (2023). "Open Source Software and Global Entrepreneurship". In: *Research Policy* 52.6, p. 104846. DOI: 10.1016/j.respol.2023.104846. URL: https://www.sciencedirect.com/science/article/pii/S0048733323001300?via%3Dihub#bb0155.

Haefliger, Stefan et al. (2008). "Code Reuse in Open Source Software". In: *Management Science* 1. URL: https://pubsonline.informs.org/doi/abs/10.1287/mnsc.1070.0748.

Varian, Hal R. et al. (Dec. 2003). *Linux Adoption in the Public Sector: An Economic Analysis*. Accessed: 2023-10-13. URL: https://faculty.haas.berkeley.edu/shapiro/linux.pdf.

Nagle, Frank (July 2018). “Learning by Contributing: Gaining Competitive Advantage Through Contribution to Crowdsourced Public Goods”. In: *Organization Science* 29.4, pp. 569–587. DOI: 10.1287/orsc.2018.1202. URL: https://pubsonline.informs.org/doi/10.1287/orsc.2018.1202.

Gallo, Katlyn (Sept. 2022). *Log4J Vulnerability Explained: What It Is and How to Fix It*. Built In. URL: https://builtin.com/cybersecurity/log4j-vulerability-explained.

Bridgwater, Adrian (2019). *The Impact Of The Tech Giants On Open Source*. Senior Contributor, Forbes. URL: https://www.forbes.com/sites/adrianbridgwater/2019/09/07/the-impact-of-the-tech-giants-on-open-source/ (visited on 09/07/2019).

Habusha, David (2019). *Personal Communication*. Vice President, WhiteSource. Taken from https://www.forbes.com/sites/adrianbridgwater/2019/09/07/the-impact-of-the-tech-giants-on-open-source/.

National Telecommunications and Information Administration (n.d.). *Software Bill of Materials*. http://ntia.gov/SBOM. Accessed: 2024-03-05.

Alvarenga, Gui (2023a). *SBOM (Software Bill of Materials)*. https://www.crowdstrike.com/cybersecurity-101/secops/software-bill-of-materials-sbom/. Accessed: 2024-03-12.

Linux Network Plumber (July 2009). *Congratulations Microsoft*. https://linux-network-plumber.blogspot.com/2009/07/congratulations-microsoft.html. Accessed: 2024-03-12.

Schwartz, Samantha (2021). *One year later: Has SolarWinds changed how industry builds software?* https://www.cybersecuritydive.com/news/solarwinds-1-year-later-cyber-attack-orion/610990/. Published Dec. 14, 2021.

The White House (n.d.). *Executive Order on Improving the Nation's Cybersecurity*. https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/. Accessed: 2024-03-12.

Hendrick, Stephen et al. (Sept. 2023). *The 2023 State of OSPOs and OSS Initiatives*. Accessed: 2023-10-13. URL: https://www.linuxfoundation.org/hubfs/LF%20Research/2023%20State%20of%20OSPOs%20-%20Report.pdf.

Larios Vargas, Enrique et al. (2020). “Selecting third-party libraries: The practitioners’ perspective”. In: *ESEC/FSE 2020 - Proceedings of the 28th ACM Joint*

Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2020 - The 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery (ACM). URL: https://arxiv.org/abs/2005.12574.

Reflectiz (Oct. 2022). *How to Automate the Software Bill of Materials (SBOM)*. Blog. URL: https://www.reflectiz.com/blog/software-bill-of-materials-sbom/.

Worthington, Janet et al. (June 2023). *The Forrester Wave™: Software Composition Analysis, Q2 2023*. https://www.forrester.com/report/the-forrester-wave-tm-software-composition-analysis-q2-2023/RES178483.

GitHub (n.d.). *Open Source Friday*. https://opensourcefriday.com/. Accessed: 2024-03-01.

Gatto, James G. (June 2019). *Open Source Software Policies – Why You Need Them And What They Should Include*. https://www.lawoftheledger.com/wp-content/uploads/sites/15/2019/06/Notes-On-Open-Source-Policies-Article-0619.pdf.

Mishra, Devsena (Oct. 2023). "What is 'Dependency Hell'?" In: *Medium*. Accessed: Insert current date here.

Springett, Steve (n.d.). *Component Analysis*. https://owasp.org/www-community/Component_Analysis. Accessed: 2024-03-12.

Haddad, Ibrahim (Sept. 2022). *A Deep Dive Into Open Source Program Offices: Structure, Roles, Responsibilities, and Challenges*. https://www.linuxfoundation.org/research/a-deep-dive-into-open-source-program-offices. Accessed: 2024-03-12.

Google (2022). *About Google Open Source*. https://opensource.google/about. Accessed: 2024-03-12.

Alvarenga, Gui (Nov. 2023b). *SOFTWARE COMPOSITION ANALYSIS (SCA)*. Accessed: 2023-10-13. URL: https://www.crowdstrike.com/cybersecurity-101/cloud-security/software-composition-analysis/.

W3Techs (Mar. 2024). *Usage statistics of PHP for websites*. Accessed: 2024-03-22. URL: https://w3techs.com/technologies/details/pl-php.