"Always Contribute Back": A Qualitative Study on Security Challenges of the Open Source Supply Chain

Dominik Wermke*, Jan H. Klemmer†, Noah Wöhler*, Juliane Schmüser*,
Harshini Sri Ramulu‡, Yasemin Acar‡§, and Sascha Fahl*†
*CISPA Helmholtz Center for Information Security, Germany,
{dominik.wermke,noah.woehler,juliane.schmueser,fahl}@cispa.de
†Leibniz University Hannover, Germany, klemmer@sec.uni-hannover.de
‡Paderborn University, Germany, {harshini.sri.ramulu,yasemin.acar}@uni-paderborn.de
§George Washington University, United States

Abstract—**Open source components are ubiquitous in companies' setups, processes, and software. Utilizing these external components as building blocks enables companies to leverage the benefits of open source software, allowing them to focus their efforts on features and faster delivery instead of writing their own components. But by introducing these components into their software stack, companies inherit unique security challenges and attack surfaces: including code from potentially unvetted contributors and obligations to assess and mitigate the impact of vulnerabilities in external components.**

In 25 in-depth, semi-structured interviews with software developers, architects, and engineers from industry projects, we investigate their projects' processes, decisions, and considerations in the context of external open source code. We find that open source components play an important role in many of our participants' projects, that most projects have some form of company policy or at least best practice for including external code, and that many developers wish for more developer-hours, dedicated teams, or tools to better audit included components. Based on our findings, we discuss implications for company stakeholders and the open source software ecosystem. Overall, we appeal to companies to not treat the open source ecosystem as a free (software) supply chain and instead to contribute towards the health and security of the overall software ecosystem they benefit from and are part of.

1. Introduction

Open Source Components (OSCs) play an important role in many companies' and software teams' setups and processes. Whether as libraries and packages included in their software, as foundation or glue for their development and deployment processes, or as part of an even longer software supply chain: Utilizing external software components as building blocks in their processes and products enables companies to leverage the benefits of Open Source Software (OSS), allowing them to focus their efforts on features and faster delivery. According to a 2020 RedHat report, 95% of IT departments and companies consider OSS as strategically important to their organization's overall enterprise infrastructure software strategy [1].

By introducing external components into processes and their stack, industry projects inherit the unique challenges and attack surfaces from open source projects: companies are including code from potentially unvetted contributors or sources and are now obligated to assess and mitigate the impact of vulnerabilities from external code included in their software. While not strictly open source, the impactful SolarWinds Orion attack wave highlighted the industry's vulnerability to compromised external code components [2]. In December 2020, cybersecurity company FireEye discovered that advanced persistent threat actors had created a backdoor hidden in a software update of SolarWinds' Orion system, affecting almost 18,000 customers worldwide [3]. Malicious actors are aware of the widespread use of OSCs in the industry and have tried to leverage this attack vector in the past: In August 2022, 10 packages on the popular Python Package Index (PyPI) were found to be malicious by Checkpoint [4]. Installing one of these packages triggered a malicious script, crawling the local browser storage for passwords, cookies, and crypto-currency wallets, extracting them via a Discord server hook. The integrity of OSCs is not only threatened by malicious external actors: In a so-called protestware incident, the JavaScript *node-ipc* library (dependency of, e. g., *@vue/cli* and the Unity game engine) was updated by the maintainer as a protest to Russia's invasion of Ukraine in early March 2022. Depending on the version and if the machine's IP matched a list of Russian or Belarusian addresses, the library would replace all of the user's system files with heart emojis [5]. This and other recent protestware incidents highlighted, that even initially well-meaning changes can be conceived as threats to the software supply chain and harm the trust in OSS.

In this work, we aim to shed light on the security challenges and considerations of companies and software teams around including OSCs in their projects and processes—by exploring industry projects' behind-the-scene processes, provided guidance and security policies, as well as past security challenges and incident handling. Our research approach utilizes 25 in-depth interviews with software developers, architects, and engineers from a diverse sample of industry projects and companies, to investigate the importance of OSCs in companies' software stacks, as well as related security challenges and considerations, guided by the following research questions:

RQ1. *"How are Open Source Components included in companies' tech stacks in terms of position, importance, and security effects?"* OSCs hold an important role in many companies' software stacks. We are interested in the specific roles of these components in the software stack, as well as if and how these components are considered in the update and security processes of the projects.

RQ2. *"What are companies' awareness, experiences, and attitudes regarding the security of including external open source code?"* Including external OSCs in industry projects introduces unique security challenges and attack vectors such as code contributions from unvetted sources. We are interested in companies' awareness surrounding the security of including external open source code, as well as their experiences with, and past challenges of, including external code in the context of security and updates. We are also interested in the companies' attitudes about including, managing, and contributing back to open source projects.

RQ3. *"If and how do stakeholders make decisions and considerations around security and trust challenges of including Open Source Components?"* The major impact of security challenges in OSCs justifies specific considerations. We are interested in measures that companies utilize to decide on including OSCs, what decisions and considerations they have in place for the external code, and which improvements and changes stakeholders consider.

This work is structured as follows: After this general introduction (Section 1), we discuss related work in the areas of dependency analyses and selection, security research with software developers, and interview studies in a security context (Section 2). We then describe our interview approach (Section 3) and highlight our findings (Section 4). Finally, we discuss our findings (Section 5) and draw a conclusion (Section 6).

1.1. Replication Package

In line with the effort to support replication of our work and help other researchers build upon it, this publication has a companion website with a full replication package and an artifact repository available.[1]

2. Related Work

In this section, we present and discuss related work in three areas: research investigating dependencies and the selection thereof, security research involving software developers and similar stakeholders, as well as interview studies with a focus on security. We also put our work into context and illustrate the novel contributions of our research.

1. https://publications.teamusec.de/2023-oakland-oss-consumers/

Dependency Analysis & Selection. Dependencies are a popular (security) software research topic, as they can hide critical attack vectors and attack surfaces. Dependency ecosystems are a common data source for measurement studies in this field, e. g., for package repositories like JavaScript's npm [6]–[11], Python's PyPI [12], [13], Ruby's gem [14], R's CRAN [15], and the wider software ecosystems like for Apache [16], Gentoo [17], Java [18], [19], or Android [20], [21]. The inclusion of third-party dependencies and the associated technical challenges have been studied and compared across a variety of software ecosystems [22]–[25]. In 2020, Ponta *et al.* presented a novel method for detecting vulnerabilities in OSS dependencies [26]. The propagation of vulnerabilities within the npm ecosystem has been studied with the help of dependency trees [27] and dependency graphs [10]. The obfuscation-resilient detection of libraries in Android apps has been advanced for in-depth analyses of apps with a focus on malicious third-party libraries and malware detection [28]–[30]. The selection of dependencies is crucial for supply chain security. However, it is also a major challenge because approaches, criteria, and metrics for good and secure choices are hard to generalize and various exist. In 2010, for example, Mileva *et al.* mined and evaluated API popularity and trends for 200 Java projects [31]. The authors demonstrate that it is possible to give adoption recommendations based on past usage trends. Similarly, libraries that are already included in a project can also be used for further library recommendations, as Nguyen *et al.* demonstrated with the library recommendation system *CrossRec* [32]. In 2020, Xu *et al.* surveyed 49 developers from GitHub and F-Droid to analyze reasons why developers replace own code with a library, i. e. re-use, or re-implement a library's functionality [33]. In 2018, López de la Mora *et al.* proposed a metric-based approach for informed adoption decision when selecting and comparing libraries [34], [35]. Kula *et al.* conducted an empirical study on library migration covering 4,600 GitHub software projects and 2,700 library dependencies, finding that 81.5% of the studied systems still keep their outdated dependencies [36].

Supply-chain attacks and vulnerabilities and challenges have been systematized [37], [38] and analyzed to inform the development of protective measures [39], [40], to improve the accuracy of vulnerability alerts [41], and to better understand the factors that influence dependency vulnerability remediation in software projects [42]. Larios Vargas *et al.* identified 26 technical, human, and economic factors that developers consider in their dependency selection processes based on 16 interviews and a survey with 115 developers [43]. Basak *et al.* investigated practices for secret management in software artifacts [44]. More recently, in two preprints, Zahan *et al.* utilized Open Source Security Foundation (OpenSSF) Scorecards, both for investigating security features in npm and PyPI, as well as their impact on security outcomes, highlighting some impactful features [45], [46]. Compared to prior work investigating dependencies utilizing measurements and systematization, we leveraged interviews to investigate in-depth the real-world selection and inclusion practices of, and experiences around, open source components in companies and software teams, providing additional and enhancing insights to previous measurement results.

Security Research with Software Developers. Research investigating security aspects with developers, architects, and engineers working on industry projects provide important insights into the security and health of the overall software ecosystem.

Past research investigated the security impact of different aspects such as decision-making [47], [48], organizational changes [49], [50], and information sources [51], [52]. Stevens *et al.* conducted a multi-stage study with 25 industry employees investigating aspects of threat modeling [53]. Assal *et al.* surveyed 123 software developers about software security processes, finding that the real issues frequently stem from a lack of organizational or process support [54]. More recently, Ladisa *et al.* introduced a taxonomy for attacks on open source supply chains, validating their taxonomy by surveying 17 domain experts and 134 software developers [55]. Similar to prior work with software developers, we consider industry developers and software teams to play an important role in the overall security and health of the software supply chain.

Security Interview Studies. A common approach for in-depth, qualitative research in the security community are interview studies. Prior interview studies were conducted to establish the security needs of expert communities such as journalists [56], editors [57], and victim service providers [58]. As part of larger studies, interviews allow insights into specific mindsets and approaches, e. g., for encryption tasks [59] or Tor usage [60]. Huaman *et al.* conducted 5,000 computer-assisted telephone interviews with small and medium enterprises in Germany, finding that security awareness has arrived in all companies [61]. More related to this work, past research has utilized interviews to gain insights into the work and tools of experts such as security professionals [62], app developers [63], and administrators [64], [65]. Specifically, Botta *et al.* interviewed 12 security management professionals, finding that the job of IT security management is distributed across multiple employees [66]. Haney *et al.* conducted 21 interviews in organizations including cryptography in products, finding a uniquely strong security mindset in those companies [67]. More recently, both Jansen *et al.* and Ghofrani *et al.* conducted smaller-scale interview studies with industry developers investigating the trust aspect of external software [68], [69]. Compared to these smaller-scale, preliminary works, our work focuses less on specific trust aspects, with our approach covering the broader topic of OSC in companies, covering real-world usage, company policies, and security considerations. Gutfleisch *et al.* interviewed developers about usability considerations in their secure software development processes, identifying a high impact of contextual factors [70]. Wermke *et al.* interviewed 27 open source maintainers about security and trust considerations in their projects, finding that the projects were highly diverse both in deployed security measures and trust processes [71]. Fourné *et al.* interviewed 24 participants from the Reproducible-Builds.org project, identifying experiences that help and hinder adoption [72]. Compared to the last two, we also leveraged in-depth interviews to gain detailed insights, but with a focus on the "other" end of the open source software supply chain, interviewing stakeholders of industry projects in the context of OSCs they use.

Overall, we leveraged 25 interviews with participants from industry projects to investigate the broader picture of OSCs in companies and software teams, covering topics including, but not limited to, real-world usage, company policies, and security considerations.

3. Interview Study

In this section, we outline the interview approach including the structure of our interview guide, the subsequent coding and analysis steps, ethical considerations, and potential limitations of our research approach. The full interview guide and codebook are included with the replication package (cf. Section 1.1).

3.1. Study Setup

To investigate security considerations and experiences around OSCs in companies and software teams, we conducted semi-structured interviews ($n = 25$) with software developers, architects, and engineers experienced in industry software projects between May and October 2022. We opted for interviews as a qualitative approach, because we wanted to focus our investigation on processes not necessarily visible on a software level and rationales, e. g., how a decision for or against including a component is made, how incidents are handled internally, or what the (potentially unwritten) policies for including external code look like. Conducting this research study as interviews also allowed us to explore participants' decisions and considerations in-depth by asking follow-up questions.

Interview Guide. We conducted the interviews with an established interview guide based on our research questions. In addition to our research questions, we also considered concepts and findings from previous and ongoing related work and adapted them for in-depth interviews. We gathered feedback from, and tested the initial interview guide with, pilot interviews in the team and with industry stakeholders from our professional network. After the initial pilot interviews, we only performed relatively minor changes: Adding a few minor follow-up questions to improve coverage of interesting areas, updating some question wording, and moving questions between interview guide sections for better interview flow. No further changes beyond minor grammatical modifications were added after the 8th interview. The full interview guide is included in the replication package (cf. Section 1.1).

Recruitment and Inclusion Criteria. We based our recruitment approach around covering a diverse set of industry projects utilizing OSCs. For recruitment, we utilized multiple recruitment channels to better reach a diverse set of companies from different historical, structural, and industry

TABLE 1. Detailed overview of interviewed software developers, their projects, as well as some project metadata. According to our interview guidelines, participants were assigned an alias and their projects' metadata was binned to preserve their privacy.

Alias	Interview			Projects		
	Duration	Codes[1]	Recruitment Channel	Position[2]	Area	Software Stack[2]
P01	46:21	31	Professional Network	Developer	Machine Learning	Python, Flask, AWS
P02	50:59	34	Professional Network	Sec Engineer	Finance, VR	JavaScript
P03	33:54	29	Professional Network	Lead Dev	Embedded	C, STM32
P04	29:02	29	Professional Network	Team Lead	Mobile	Android
P05	39:19	31	Professional Network	Lead Engineer	Framework	Python
P06	30:21	26	Professional Network	Developer	Industry	Java, Spring
P07	38:20	25	Industry Expert	Senior Engineer	Finance	Node.js, SQL
P08	54:15	24	Industry Expert	Lead Dev	Web Apps	PHP, Laravel, MySQL
P09	35:17	36	Industry Expert	Lead Dev	Web Apps	Angular JS, ASP.NET, Python, C#
P10	40:33	27	Industry Expert	Architect	Various	Java, Maven, Terraform
P11	41:01	26	Industry Expert	Senior Engineer	Enterprise Apps	Java, Node.js, Angular JS
P12	52:04	27	Industry Expert	Founder	Enterprise Apps	Java
P13	25:20	30	Industry Expert	Developer	Web Apps	PHP, WordPress
P14	36:50	28	Industry Expert	Developer	Backend	React
P15	49:51	19	Industry Expert	Consultant	Various	Java
P16	49:24	35	Industry Expert	Developer	Finance	Angular JS, Vue.js
P17	26:25	30	Industry Expert	Architect	Various	ASP.NET, Angular JS, React Native
P18	39:25	32	Industry Expert	Developer	Mobile	Android, Spring, Angular JS
P19	27:39	30	Industry Expert	Expert, Architect	Embedded	Terraform
P20	29:04	33	Industry Expert	Developer	Enterprise Apps	JavaScript, Ruby on Rails
P21	33:29	22	Industry Expert	Developer	Health & Wellness	Java, PostgreSQL
P22	51:04	28	Industry Expert	Team Lead	Web Apps	JavaScript, React, Node.js
P23	28:09	25	Industry Expert	Developer	Web Apps	.NET, C#, React
P24	43:04	28	Industry Expert	Developer, Auditor	Mobile	C++, C-Basic, C#, Flutter
P25	33:56	30	Industry Expert	Developer	Blockchain	React, Python

[1] Total number of codes assigned to the interview after resolving conflicts. [2] Based on self-reporting of participants and binned to preserve their privacy.

contexts. This included recruiting expert talent via Upwork and our professional network:

1) *Industry Experts.* For recruiting expert talent, we turned to Upwork, a platform for professional developers and freelancers. We posted a job posting for our interviews and specifically selected participants based on their experiences working in company projects utilizing some form of open source, aiming for a diverse sample with a broad coverage of the industry.
2) *Professional Network.* In addition to Upwork, we enhanced our sample with professionals from our own network, specifically targeting software solutions that are less commonly encountered in industry but still play important roles, such as embedded hardware or research software projects.

See also Table 1 for an overview of interviewed participants and corresponding recruitment channels. Due to the previous filtering, we did not require any additional eligibility criteria from our participants beyond stating that we were looking for professionals working on industry projects utilizing OSCs. In total, we recruited 25 participants from equally as many distinct companies and projects. As compensation for their valuable time as domain experts, we offered each participant $60 or the equivalent value in local Amazon vouchers.

Interview Procedure. We conducted the 25 interviews either in a solo interviewer or lead and backup interviewer configuration. We chose the lead and backup interviewer setup so that the lead interviewer can fully concentrate on asking questions and listening to the interviewee, and the backup interviewer could ensure that no questions are forgotten, ask additional follow-up questions that emerge, or take over in case of any connection issues. We conducted all interviews virtually; mostly via our self-hosted *Jitsi* instance, or any other tool of the participant's choice (e. g., *Zoom*, *Google Meet*, etc.). We advertised the interviews with a duration between 35–45 minutes depending on answer duration and scheduled one hour interview appointments for some time to spare. Overall, the median duration of the actual interview part, excluding short introduction, consent gathering, and debriefing, was 38:20 minutes.

In general, the interviews were based around non-leading, open questions, allowing interviewees to elaborate their thoughts and answers. Each interview section started with a general question, allowing participants to freely state what they had on their mind. Only if specific points were not already addressed by that time, we asked more specific sub-questions as follow-ups. All interviewers were instructed not to prime participants through the questions and not to impart any sense of judging, e. g., regarding specific OSC choices or security practices.

3.2. Interview Structure

We report on the structure of the semi-structured interviews below and in Figure 1. The interviews were structured in six main sections consisting of one to four opening questions, corresponding follow-up questions, and sometimes

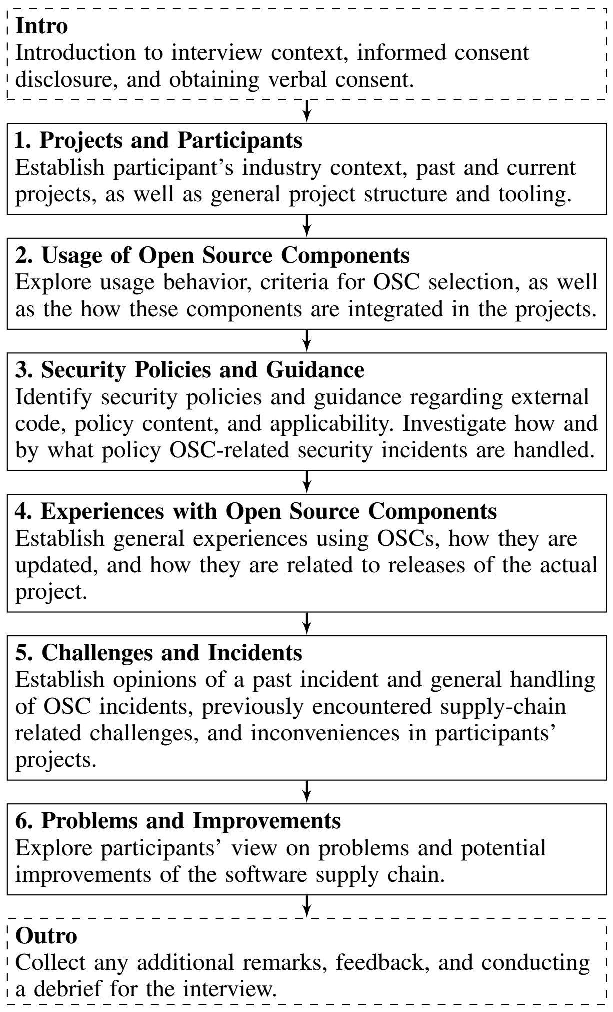
Intro
Introduction to interview context, informed consent disclosure, and obtaining verbal consent.
1. Projects and Participants
Establish participant's industry context, past and current projects, as well as general project structure and tooling.
2. Usage of Open Source Components
Explore usage behavior, criteria for OSC selection, as well as the how these components are integrated in the projects.
3. Security Policies and Guidance
Identify security policies and guidance regarding external code, policy content, and applicability. Investigate how and by what policy OSC-related security incidents are handled.
4. Experiences with Open Source Components
Establish general experiences using OSCs, how they are updated, and how they are related to releases of the actual project.
5. Challenges and Incidents
Establish opinions of a past incident and general handling of OSC incidents, previously encountered supply-chain related challenges, and inconveniences in participants' projects.
6. Problems and Improvements
Explore participants' view on problems and potential improvements of the software supply chain.
Outro
Collect any additional remarks, feedback, and conducting a debrief for the interview.

Figure 1. Overview of the interviews' flow and topics. After introducing each section with a general question, we followed-up with specific questions (if not already covered). Due to our semi-structured interview approach, participants were allowed to diverge from this flow at any time.

additional nudges or explanations. We also report on the results from the interview in sections corresponding to the interview guide's question sections (cf. Section 4).

Before starting the interview, we provided participants with a general introduction of ourselves and our research project, followed by an explanation of our goals, the interview process, and the interview's role in that process. We specifically affirmed to participants that participation in the interview is voluntary, that they could skip any question for any reason, that we were not judging their projects in terms of security or privacy, and that we were also very interested in their personal thoughts and opinions about processes. We guaranteed full de-identification of any quotes we might use and offered to send participants a preprint of the potential scientific publication based on their interviews.

After answering any remaining questions and obtaining consent for data handling and recording from the participant, we started a recording and began the actual interview with the following structure:

1. Projects and Participants. In the first interview section we asked our participants to describe their projects, their relation to it, as well as project structures and tooling. This section intends to both ease nervous participants into the interview and to establish some initial context about the participant and their projects. Specifically, we prompted for project context and structure, team size and tools, as well as for the development process and stages. We report these results in Section 4.1.

2. Usage of Open Source Components. Both the *"Usage of Open Source Components"* and *"Thoughts about Open Source Components"* sections investigate the usage of open source components in our participants' projects. Specifically, we were interested in the technical implementation and processes, selection and exclusion criteria for open source components, as well as whether they contributed back to open source projects in some way. We report these results in Sections 4.2 and 4.3.

3. Security Policies and Guidance. Our third block of questions covers security policies and guidance for including external code like OSC in projects. We asked about company policies and project-specific guidance or documentation for the inclusion of external code, and the participants' personal opinions and wishes regarding these. Additionally, we investigated the general processes and policies for security incidents in external components. We report these results in Section 4.4.

4. Experiences with Open Source Components. The fourth section focuses on the participants' personal experiences with OSCs in their projects. Our questions covered aspects such as the developer experience of using OSCs and if components had to be customized for the projects. We also investigated the update, release, and deprecation procedures of the projects in the context of external components. Lastly, we asked whether our participants would use the same components again and why. We report these results in Section 4.5.

5. Challenges and Incidents. In the fifth interview section, we are interested in specific OSC-related security incidents and inconveniences experienced by our participants in the past. To ease our participants into this sensitive topic, we asked them about their opinion regarding open source software supply chain security of the "*node-ipc protestware*" incident from March 2022 [5]. We inquired about participants' opinion of the incident, as well as for their strategies to deal with similar incidents in their project. We then specifically asked them about any OSC-related security incidents or inconveniences their projects might have encountered in the past. We report these results in Section 4.6.

6. Problems and Improvements. In our final interview section, we investigate our participants' opinion of their projects' security, as well as problems they see with the current software supply chain and their suggested solutions. We report these results in Section 4.7.

Following the interview sections, we asked our participants for any additional insights and aspects that we might have missed or they wanted to talk about. After completing the interview, we thanked them for their valuable time and offered them an opportunity for questions and comments, concluding the interview with a debriefing.

3.3. Coding and Analysis

For our evaluation of the interviews, we recorded the audio of interviews digitally, removed identifying information from recordings, transcribed them via a GDPR-compliant service, and manually reviewed all transcripts for potential transcription mistakes. We analyzed all interview answers in an iterative semi-open coding approach [73]–[75]. All researchers together established an initial codebook based on the interview guide and interview impressions. Five researchers then iteratively coded the interviews according to the codebook in multiple rounds, resolving conflicts by consensus decision or by introducing new (sub)codes after each iteration. We continued with our iterative coding approach until no new codes or themes emerged [76], [77]. Our approach does not necessitate the reporting of inter-coder agreement, as each conflict is resolved when it emerges (resulting in a hypothetical final agreement of 100%) [78]. In total, we assigned 715 codes after resolving, resulting in a median of 29 codes per interview. The final codebook is included in our replication package (cf. Section 1.1). As part of discussing our results, we report on some counts. We want to highlight that counts from a qualitative interview study with a sample selected for diverse background are not intended to be representative counts for the wider developer population, but are included to give some general idea about the distribution of codes and to highlight especially prevalent or underrepresented themes in the interviews.

3.4. Ethical Considerations & Data Protection

This interview study was approved by our institutions' Institutional Review Board (IRB) as well as Human Subjects Review Board (IRB equivalent). Our study was modeled after the ethical principles for research involving information and communication technologies outlined in the Menlo report [79]. The research plan, study procedure, and all involved research parties adhered to the strict German data and privacy protection laws as well as the General Data Protection Regulation (GDPR). Before signing up for interviews, we provided participants extensive information about our study procedure and data handling, encouraged them to get informed before making a decision, and offered to answer any questions they may have had. We emphasized to participants that they could skip any question for any reason such as not knowing an answer, not wanting to answer, or not being allowed to answer, as well as that they could drop out of the interview at any time. We provided participants with a preprint of this work before publication, allowing them to request changes or to correct misunderstandings. To compensate our domain expert participants, we offered them $60 or the equivalent value in local Amazon vouchers.

All data was collected, handled, and stored in compliance with the EU's GDPR. In accordance, any personally identifiable data was stored in a secure cloud collaboration software, encrypted at rest and in transit. For transcribing the interviews, we commissioned an EU-based, fully GDPR-compliant transcription service.

3.5. Limitations

A number of limitations typical for this kind of interview study apply to our work, including potential over- and under-reporting, self-reporting, recall, and social-desirability biases, as well as sampling bias. Our sample is a convenience sample which may not be representative of the larger population of developers working on industry projects utilizing OSC. Experts who agreed to participate in our study might be more or less open source or security-oriented than those did not sign-up for an interview. We conducted our interviews in English, so we cannot provide insights into non-English-speaking industry projects. As English is the *de facto* "working language" of international software projects, we consider this a negligible drawback that still allows us to reach a meaningful set of developers. Since questions about security practices and incidents can be considered sensitive, we attempted to mitigate social desirability bias by emphasizing that we were not going to judge the participants or their answers in any way but were genuinely interested in their processes and opinions. We also reminded them that they could skip questions as desired and for any reason.

4. Results

We report and discuss results for 25 semi-structured interviews with software developers, architects, and engineers. In our reporting, we mostly adhere to the structure of the interview guide described in Section 3.2 and summarize our key findings after each question block. We report participants' quotes as transcribed, with de-identified information, minor grammatical corrections, and omissions highlighted by brackets ("[...]").

4.1. Projects and Participants

In total, we interviewed 25 valid participants, reporting on their projects and background in this section and Table 1. We only report binned project metrics to preserve both our participants' and their projects' privacy. Due to our recruitment approach aiming for a high diversity in projects, our participants reported a wide range of projects and backgrounds, ranging from web applications, over embedded devices, to scientific computing frameworks.

As the vast majority of our participants (23) had worked on multiple projects in the past, we encouraged them to highlight aspects of their projects as they saw fit. The majority of our participants (22) worked or had worked on projects in teams, specifically with two to five (9) or more than five (13) developers. About half (13) mentioned having worked on projects with multiple teams, e. g., "*We are a very flat team, so basically everyone is on the same level. We discuss things together and we work together. We have a development team for back-end code, for front-end code, and the design team.*" (P13) We were also interested whether this included specific teams or members with a security background: 11 participants mentioned security-specific roles on their projects, e. g., "*Yes, we have online software security engineer or cybersecurity engineer [. . .]. Then you have more dedicated roles for the information assurance processes and some of the other cloud based [services].*" (P19) These 11 also include security-specific roles provided by clients, e. g., for P12: "*Actually, most of the time, our clients are enterprise clients and we hand over the codebase to them and then their security team.*" (P12) About half of our participants (13) specifically mentioned not having someone with security-specific background or experience in the loop, e. g., "*No, I haven't worked with any company [that] had something like [a security-specific role].*" (P18)

Overall, we found our participants to be quite knowledgeable about their projects, with many years of experience in different areas of software development. This allowed us in-depth insights into the considerations around OSCs in their projects, a goal we hoped to archive with our recruitment strategy. Based on our findings, an interesting area for future research could be how security processes differ between the different industry areas.

> Summary: Projects and Participants. The majority of our participants had worked on multiple projects in a diverse set of software areas, and in different team configurations and sizes. Only about half mentioned security-specific roles in the development loop.

4.2. Usage of Open Source Components

In this interview section, we were interested in the general usage and selection criteria of OSCs (for specific experiences with OSCs, see also Section 4.5). All 25 of our participants mentioned using OSCs in their projects (unsurprisingly, as we specifically selected for this), e. g., "*Every solution that we have built, we heavily use open-source components.*" (P10) Some participants even voiced a philosophical attachment to the idea, e. g., "*We are using a lot of open-source things because, by philosophy, we like to embrace open-source community*" (P07), including perceived benefits such as reduced maintenance burden: "*The software is well maintained, and we don't have to focus on maintaining the work. We can just use them, and if we see any problems, we can contribute to this and we don't have to do any in-house maintenance.*" (P07) For pulling in OSCs into their build processes, our participants have a number of different approaches, often relying on the (semi-)official package repositories such as PyPI or npm, e. g.,

> "*So for builds we use npm and also some other package managers. Usually it's a mixed project with different code bases, and we pull the packages directly from the package managers. I don't think we currently have any that are not managed by package managers, which is great.*" — P02

Aside from package repositories, some participants mentioned directly pulling from repositories if there are no other options available: "*[For] some of the dependencies, if either the current version is not maintained [on PyPI] or for some machine learning tools [. . .] we pull them directly from the Git repo for them.*" (P01) Others mentioned directly pulling from GitHub as a potential security concern:

> "*I think some components are fairly secure because we draw from sources posted and maintained by hardware manufacturers. [. . .] These are open source components, but they have major corporate backing behind them, and we're not pulling it from GitHub.*" — P04

Some participants specifically mentioned configuring and modifying OSCs to fit their needs and the requirements of their projects: "*We pull them in, and then all of these open source components, we take them and then we do a bit of work on it ourselves. So, for example, we run packet managers and we add an external code.*" (P17)

Updates & Releases. As for keeping their OSCs up-to-date, our participants seem to follow the same pattern for pulling in components, i.e., relying on the package management tools of their software stack, including tools like *npm audit*: "*We even have a mechanism that lets the build fail if there is a component that could not be updated if an update is available or if there is a vulnerability reported by npm, for example.*" (P02) Others routinely revisited included components: "*It's some of the external dependencies that, yes, can go out of date or can disappear, that are more of a cause of concern. That's why we periodically revisit that whenever we upgrade or whenever we make a new release.*" (P04) Outside of the build and update process, some participants mentioned to keep track of included components via their package managers, but only one specifically mentioned maintaining a Software Bill of Materials (SBOM): "*We do regular scans, we build Software Bill of Materials [. . .] and we put those Bill of Material files into Dependency-Track which is free software, open source provided by OWASP that we also use for vulnerability management and overview.*" (P02) Both tooling and research around the recently established SBOM format might be a promising research opportunity.

Of our 25 participants, 14 mentioned that they (at least partially) use internal mirrors for pulling software into their build processes, e. g., "*We do use internal mirrors mainly for speed and convenience, especially large code bases. [. . .] Usually, when we clone from those internal repositories, we're going to use fixed commits from it, so it makes development a lot easier,*" (P04) and three participants mentioned other solutions like local build caches: "*We have a local cache. So we try to have everything pulled only once and put it in our cache, but sometimes we get upstream changes, so we pull it from there.*" (P03)

Selection Metrics. We were interested in what metrics and criteria our participants use for deciding on and selecting

OSCs. Most commonly mentioned metrics included: some form of popularity measure like downloads or GitHub stars (16), a large and active community (11), specific features (10), and activity measures like commit frequency and recent releases (10). We were also interested in what criteria would exclude an component from being used by our participants. Most common exclusion criteria included: the project being visibly inactive (5), a low number of contributors (4), and specific company policies (3). On the security side, participants mentioned looking for a positive security history (8) and exclusion based on vulnerable or malicious code (3), e. g., "*if the vulnerability score is ridiculously high, obviously, then we would not allow that [component]. Also, if certain vulnerabilities exist for functionalities that we want to actually use, if those are compromised, then obviously it's not a good choice.*" (P02) Lastly, we were interested in whether our participants had used or heard of recently emerging automatic metric tools before, such as the OpenSSF Scorecards for repositories. The majority (17) had not, and only four had heard of (but not used) the OpenSSF Scorecards project specifically: "*Yes, I have heard of those, but we have not used them yet.*" (P02) For other metric tools, e. g., P12 mentioned: "*We are using some integrated code quality tools, and there are some predefined, maybe hundreds of rules to check the quality of the code. These tools also provide a feedback on code security.*" (P12)

Overall, our participants' selection metrics and criteria seem to focus on quickly accessible numbers and facts, such as downloads, GitHub stars, and time since the last release. Understandably so, as there a often many different open source packages to be considered for each use case.

> Summary: Usage of Open Source Components. All of our participants included OSCs in their projects. About half maintained internal mirrors or caches for their builds. Common selection and exclusion criteria included easily visible metrics like activity, number of contributors, or GitHub stars.

4.3. Thoughts about Open Source Components

Aside from our participants' usage of OSC in their projects, we were also interested in their thoughts about supporting the open source ecosystem, company policies prohibiting packages, and which metrics they would *like to use* for selecting components.

We asked participants if their company contributed back to open source projects in some form including pull requests or issues, which 14 do and five would like to, e. g., "*We rarely submit PRs, but we submit issues, and if we find some bugs or enhancements regularly, because, often, there are some bugs, and also, we found a few security-related, actually.*" (P01) Some participants even mentioned contributing back as a company policy: "*We also do heavy contribution on anything that is being leaky, or if something is not looking right, we always create issues. This is something that we also have in our company policy: Always contribute back.*" (P07) Sometimes, this contribution was out of necessity: "*[A feature] never made to production because the [open source] project went dead completely. When we were thinking about the operation of the new release, we basically had to take over, fork it, and then implement that feature on our own.*" (P10) Some participants suggested that their management or legal departments do not fully understand the open source ecosystem, e.g., "*I think the responsible people just didn't understand the whole scope of OSC options that a developer has, because they're mostly managers and legal people, and they don't have so much insight in technical stuff.*" (P16)

For the three cases where company policies prohibit specific packages, reasons often involved the package's license: "*Many open source projects are using different licenses, some of the licenses are not okay for big projects [. . .] we check if this package is using restricted license [. . .]*" (P24) and "*There are no rules, except that the license has to be compliant with what we do.*" (P01) Some mentioned that they include their clients in these decisions: "*There are certain exclusion criteria based on our customers' concerns. For example, if they don't want us using products from certain companies, then we don't do that.*" (P04) We were also interested in what selection criteria our participants *would like* to use if they could. The wishes included specific (free) software solutions:

> "*I think there are many interesting software solutions as far as I could see. So most of them, obviously, combined with costs. I saw a lot of software that is supposed to help with managing third party vulnerabilities and so on and used for scanning. It's always a question of the price and most of the time I would say it's not really worth it.*" — P02

as well as security-focused metrics: "*It would be nice to have some security metrics. How much security leaks [the project] has, or in what time frame it will be fixed, or what time frame fixed for last issues.*" (P11) Based on our findings, we suggest developers might require metrics and tools that are still simple and free, yet cover more aspects of an OSC than just popularity.

Overall, our participants seemed to have a very positive attitude about open source projects and are aware of the importance of contributing back.

> Summary: Thoughts about Open Source Components. Most projects contributed back to open source projects in some form or would at least like to, with some participants suggesting that their management or legal departments do not fully understand the open source ecosystem. Some participants mentioned that their company's policy prohibited them from using certain open source packages, mostly due to non-permissive licenses.

4.4. Security Policies and Guidance

In this interview section we were interested in the policies around OSC usage in projects, as well as provided guidance and documentation for external components. More than half

(16) of our participants mentioned some form of company or team policy for including external code in their projects. These policies range from quite strict: "*Every single third party library that is used, installed, involved in our projects is vetted and must be approved.*" (P02), over somewhat more lenient: "*We can use anything that is signed off by our CTO and our project lead. Like any piece of code that has been vetted by them, we are free to use it on our projects.*" (P07), to fully placing trust on the individual: "*[The policy is to] make sure that the plugins or open source components that we use are still updated and they're still supported. We wouldn't want to include something very old, but it very much depends on the developer to make sure of that.*" (P13) This number also includes more or less informal policies that are nonetheless applied by the developers of the project, such as: "*Not on a systematic basis, but most of the time my team if we are using a new library, that we are going to use it for the first time, we are actually checking it in vulnerability databases.*" (P12) Participants also mentioned policies influenced by external laws and standards such as HIPAA: "*If we are including something which is not self-hosted, then we have to take [HIPAA] into consideration with what they claim happens with the data on their end.*" (P17) as well as ISO: "*In some [client] companies, [policy] follows the standard security programs like ISO and things like that.*" (P10) Other policies are less concerned with actual security and more with copyright, e. g., "*at [company], we were allowed to use things like npm and Angular, but we had to basically extract all the legal stuff, so the licenses basically, compile a list and give it to them.*" (P16)

We were also interested in what our participants wished to see in a (security) policy for external code, which included some more general advice "*You need to have clear guidance on how to select packages, which quality, how you would define the quality of the package.*" (P16) as well as specific security considerations such as: "*[...] doing searches to see if the software that we're considering using has had any prior issues, whether it's security issues, whether they've been disclosed through penetration testing or through some other means [...].*" (P09) Both P17 and P18 mentioned why such a policy might not be relevant or even a good idea for every project size: "*For small projects, I don't think those policies would help because they would create more governance and more red tapes.*" (P18) and

> "*With the size of our team I don't believe [a policy] is really needed, because if a developer wants to include a component, the code is generally reviewed before it goes in by people who would be in the know whether or not this component can be included.*" — P17

As part of policies, we were interested in how a potential security incident in an external component would be handled, by what policy, and by whom (for actually encountered incidents, see Section 4.6). Only six of our participants mentioned that they would involve or hand a hypothetical incident to a security team, e. g., "*We'll tell [the security team], hey, this needs to be patched pretty soon, and if it doesn't happen then we escalate it to the management above it, and we exert pressure on both ends.*" (P04) P18 provided us some insights on why a security team does not necessarily make sense for every company structure:

> "*There's no security team that specifically would do that because if there is a team that exists that only does that, they would probably just sit on their hands with nothing to do a lot of time, and this team needs to work across different projects across the company, then that they can handle that because the different projects in a company probably have a lot of languages, different languages, different frameworks. There's no way either they will know enough to poke in the project and respond fast enough, or there's no way the company is willing to pay money for that for a team that doesn't produce anything.*" — P18

We were also interested in whether companies had included a disclosure policy for security issues. Only seven participants mentioned something akin to a disclosure policy for the public or their clients, e. g., "*Yes, we do. Well, a policy for coordinated disclosure [...]. Internally, we try to fix it as quickly as possible and only then disclose it.*" (P05) Other companies appear to be in the process of implementing a disclosure policy, e. g., as mentioned by one participant: "*I guess the organization that kind of writes the standards that we follow is trying to adopt widespread disclosures of historical and current cybersecurity threats, but at the moment I have not seen any of those come out yet.*" (P19)

Overall, most participants mentioned some form of policy or common best practice in their teams, although some of the company policies seem more focused on fulfilling external standard or law requirements instead of ensuring the security of included external components.

Guidance & Documentation. A small majority of our participants (14) did mention not having specific documentation or guidance for OSCs in their projects. As reasons for not providing specific documentation, participants mentioned extensive approval processes and sufficiently experienced developers:

> "*It sounds like it would be just something to add to the pile because we already have approval processes and usually our developers are at least somewhat experienced and they have seen many things before [...] For other firms, for other companies, it definitely might make sense.*" — P02

Reasons for providing documentation included easier onboarding and supporting new developers by providing them with wrapped, documented versions of OSCs: "*We document [wrapped OSCs], and we present these in-house made components to our new developers with a good documentation and they know how they operate.*" (P12) Our participants seemed to agree on the usefulness of having documentation for these components and processes, e. g., P13 said: "*But [having some documentation for included OSCs is] a very good point. We should definitely have something like that because it's not always going to be me.*" (P13)

Overall, our participants appear to provide documentation for included components based on their given requirements and team context, with accessibility for mostly new developers in larger teams on the one side of the spectrum, to experienced developers with other processes in smaller teams on the other side.

> Summary: Policies and Guidance. Most projects had some form of company policy or at least best practices for including external code. Relatively few had dedicated security teams or a disclosure policy. Perspectives on providing documentation appear to depend on the team context, with larger documentation support being seemingly correlated with larger team size and the number of less experienced developers.

4.5. Experiences with Open Source Components

Aside from general usage and policies for external code, we were also interested in participants' past experiences with OSCs. Overall, our participants voiced a positive to very positive opinion about their experiences with the open source ecosystem, e. g., "*[i]t's great. It's a very vibrant ecosystem and you have a plethora of options to use,*" (P16) "*I would say pretty good. I like [the open source ecosystem] a lot. It's really easy to have issues and get them resolved,*" (P23) and "*open source technology and components are very attractive. They have to be because if not, you're not going to use them.*" (P15) Multiple participants mentioned a very important or key role of OSS in the overall software industry, e. g.,

> "*I think open-source components or open-source software in general has [a] very important role in overall software development industry. Whether companies are developing commercial solutions or building open-source or free solutions, they play a very important role.*" — P10

as well as: "*So [the ability to talk with maintainers] is why I feel that they really help us to accelerate our process of development and are pretty much a cornerstone of the software industry today.*" (P17) A common theme for the usage experience of OSCs was the friendliness and openness of the open source communities, e. g., for questions "*[‘open-source project’] says if we find any question, there is always a community back there to answer whatever we need,*" (P07) and for better understanding the structure:

> "*[. . .] when we talk to them and try to understand why something is built a certain way or not, most of them will be open to sitting down and having a discussion, and even in cases allowing us to help them change something for the better or for a completely new feature.*" — P17

This theme also included the ability to easily file issues (and get them resolved), e. g., "*[t]here are always issue trackers where you can flag any problems you have. I think I went into one case where we didn't have to put the issue up ourselves because that was already flagged by other users of the component*" (P06) and "*If there is a breaking change, we don't have to think about it, we just have to create an issue on the parent or the maintenance repository and they take care of making this in the next version.*" (P07) Further positive attributes of OSCs mentioned by our participants included speeding up development of their projects, e. g., "*[open source software] has allowed us to develop much quicker or develop applications more quickly using a lot of open source tools as part of the overall application, so I'm a big proponent of open source software*" (P19) and overall good code quality: "*So I have always particularly really liked the open source industry and what they provide, because if you go to see the code quality in most of these projects it is really good and they do cover a lot of use cases.*" (P17)

Importance of Documentation. We specifically inquired about the setup experience of open source components, and multiple participants mentioned that they see a good documentation as quite crucial for a good experience when using OSCs, e. g., "*I remember when I was junior or new in this open-source side [. . .] It was quite hard to set up or test or check or find the documentation, etc., but when you get used to it, it's mostly intuitive*" (P11) and "*It depends on how good the documentation is [. . .] Some people just write terrible documentation or they just don't write it at all. I think depending on how good they are, that can make your experience either very good or very bad.*" (P23)

Our participants seem to be divided on the actual state of documentation for OSCs, with some negative experiences on the one side, e. g., "*[m]y general experience with this technology is that there's a lack of documentation sometimes [. . .] there's more documentation and examples from the private software because of course, you're paying for the documentation as a client*" (P15) and some positive expressions on the other side, e. g., "*[i]f those are popular components, they're usually very comfortable to use because they have examples on their website. They have a little demo version [. . .] and [you can] even edit the code on it and then just copy and paste it into your project.*" (P25) As mentioned by P25, the quality and available documentation for OSCs appears to be often directly correlated with the popularity of the project, with more popular projects likely having more maintainer-hours available for creating documentation.

Customization and Using Components Again. More than half of our participants (14) mentioned that they had to customize an OSC for their projects beyond basic configuration changes, e. g., "*Yes. There's been times where contributing or even bringing in our own package, there's been a few times where I've forked and customized the open source repository, too.*" (P20) Participants also mentioned contributing back some of their customization and improvements to the open source projects, e. g., "*We've also contributed back to the code base or the open source project to try to get changes implemented as well.*" (P19) When asked whether they would select the same components again for a project, 17 responded positively, e. g., "*we are using our popular frameworks, our popular open-source components again and again and again. We have already set up a documentation for that.*" (P12) Of the remaining, six responded somewhat negatively, e. g., "*I think we could do with a few dependencies less because they are not really critical and they just add a nice-to-have feature [...] Some dependencies were pulled in that if starting over, I would probably try to avoid.*" (P05)

Overall, our participants had quite positive experiences with OSCs. Their highlights include, among others, the ability for fast iteration in their projects, the lessened maintenance burden, and the general openness of the communities and code, allowing them to understand, modify, and contribute to the open source projects that our participants utilize in their software. We see promising research venues in what constitutes a high-quality open source documentation and how to best support the customization of components without sacrificing security.

> Summary: Experiences. Our participants mentioned almost exclusively positive experiences, although some highlight the varying quality of documentations. Mentioned positive attributes included the ability to open issues (that get resolved) and the ability to directly talk to maintainers. Most of our participants would select the same components in some form again, given the choice.

4.6. Challenges and Incidents

Almost all of our participants (24) reported to have encountered some form of security challenge or annoyance related to OSCs in the past. Our participants mentioned challenges related to updates: "*One day, all of a sudden, the system stopped responding because the PHP updates didn't follow in that particular package,*" (P08) as well as out-dated and potentially vulnerable components. Another common theme was OSCs being no longer maintained or deprecated, e. g., as described by P17: "*Yes, dependency is no longer maintained is a big challenge*" (P17) but they also highlight their way forward of forking or looking for alternatives: "*At that time we will start maintaining it ourselves privately, or else see if somebody else has started a version two.*" (P17) Other participants mentioned that they updated their development process when they became aware of prevalent incidents (but were not affected) in the open source ecosystem, e. g.,

> "*I believe it is also was a Node.js developer who deleted all their repositories [...] and that is when we implemented cache for everything that we have a local copy for every open source component we are integrating in our build chain to be locally available.*" — P03

The participant is likely referring to the "*left-pad*" incident from early 2016, which involved a maintainer deleting their popular npm packages, including the widely-used "*left-pad*" package included in, and thus breaking, many other npm packages [80].

Incident Opinion and Strategy. To investigate our participants' strategies for handling trust incidents, we introduced and asked them for their opinion of the March 2022 "*node-ipc protestware*" incident [5]. In this protestware incident, the JavaScript node-ipc library was updated by the maintainer as a protest to Russia's invasion of Ukraine to, depending on library version and IP address, replace all of the user's system files with heart emojis. The majority of our participants had a mostly negative opinion of the incident (16), followed by a neutral opinion (6), and no opinion (3). No participant had a mostly positive opinion of the incident. Negative opinions mostly focused on the potential damage done to trust in the open source ecosystem and the potential to harm bystanders, e. g., "*I don't think that's appropriate when we're talking about security and trust [...] I don't really consider that inclusive of all people trying to use open source. Sorry, I don't really agree with that,*" (P23) the overall malicious look of the changes: "*That is just straight up malicious, that is a very black hat thing to do, and that should not even have reached a package manager,*" (P02) as well as the overall damage to OSS's reputation: "*It's bad for reputation of open-source software, but these things happen in commercial software also. [...] Some people want to use them for, as you said, for protesting purposes and some people want to use them for malicious activities.*" (P12)

With this recent incident as background, we asked our participants, what they would do if one of their projects depended on this package and how their general strategy for incidents would look like if a component or maintainer lost their trust. Most participants mentioned that they had not encountered something like that before, e. g., "*No. We had never had this incident or something like that, so we never thought about what we should do if this ever occurs.*" (P03) Common strategies for handling such incidents included finding an (open source) alternative, e. g., "*If we lose trust in a component, we'd also try to find an alternative. I guess it's a trade off, [if] it's the only alternative and we really need it, then we would have to think about how to make it more trustworthy or maybe contribute upstream,*" (P05) or assessing the damage first before taking any further steps, as mention by P12: "*Try to minimize the bad effects and try to contain the bad effects. Then I can maybe complain about the reputation of open-source software, but my first priority is to go ahead and fix the issue if it affects us.*" (P12)

Overall, most participants mentioned not having considered or encountered such an incident before. In general, their first strategy would consist of either finding an alternative, stepping up and forking the project, maintaining the project internally, or assessing the damage first before any further steps. Providing tooling and strategies that support developers in handling such incidents present a promising opportunity for both researchers and industry.

> Summary: Challenges and Incidents. Almost all of our participants had encountered (security) challenges or inconveniences related to OSCs, often mentioning broken updates and vulnerabilities in (out-dated) components. Our participants had mostly a negative opinion of the "*node-ipc protestware*" incident, mainly due to harming the trust in the open source ecosystem. Most did not mention a specific strategy for reacting to such incidents and would

generally look for alternatives.

4.7. Problems and Improvements

In the final question section, we asked our participants what they think the perceived security of their projects is (both by internal and external actors), as well as how they would like to improve the software supply chain security of their projects. Regarding the perception within their team or company, seven mentioned a mostly positive perception regarding their security, four a mostly negative, with the rest reporting either a neutral (5) or no perception. Regarding the perception of external actors (e.g., their clients, their users, or the public), nine mentioned a mostly positive perception regarding their security, zero a mostly negative, and again the rest reporting either a neutral (3) or no perception.

Improving Security. As for suggestions for improving the software supply chain security of their projects assuming no limitations, we roughly sorted our participants' ideas by theme, with the most mentioned including the auditing of their dependency graph and the code of external components (8), e. g., "*It would be nice to have independent audits of everything that we use, that way, we can have some level of assurance that at least the software that we're using or components we're using meets some particular standard*" (P09) and in general more developer hours (3) for testing and securing their projects or adopting OSCs, e. g., "*[...] I would like to have enough developers that we do not have to go through some of those dependencies which are not highly rated on GitHub or which are nearing the end of their maintenance lifecycle, and be able to develop those in-house.*" (P17) Other ideas for improvement included hiring a dedicated security team (2): "*If I have unlimited money, then a security team would be fine, but that's not a reality in most enterprise*" (P18) or establishing a set of best practices and documentations for open source communities (2): "*I think having some set of best practices out there that is more widely accepted among the open source development community and I guess rigid guidelines in such a way would improve what we use it for and how we use it.*" (P19) Another suggestion was the creation of a foundation or entity that could verify the security of OSCs:

> "*[T]here can be a security foundation that can offer this analysis and certification for the open source if you pay. I can be more comfortable or more confident about the technology that I'm going to propose to my boss or to the client. I can say, hey, this software is open source, but it has already been tested by this other open source foundation, but focus on security.*" — P15

Such a recently formed organization is the OpenSSF, which aims to improve open source software security through a collaborative effort—potentially highlighting a need to raise even more awareness for such efforts.

Overall, our participants' ideas for improving the software supply chain security of their projects mostly centered around having more developer-hours or tools to audit included components, as well as general security checks and penetration tests of their projects and OSCs. Providing and enabling the tooling for both auditing and testing OSC provide an opportunity for both researchers and industry going forward.

> Summary: Problems and Improvements. If they had an opinion about it, most participants thought that their projects' security is perceived as positive, both by internal and external actors. For improving the software supply chain security of their projects, participants often suggested manual and automatic audits of code and dependencies.

5. Discussion

In this work, we qualitatively investigated the role and importance of OSCs in companies and software teams, as well as the related security challenges and considerations, by conducting 25 in-depth interviews with software developers, architects, and engineers to answer the following research questions:

RQ1. *"How are Open Source Components included in companies' tech stacks in terms of position, importance, and security effects?"* Our participants mentioned OSCs in many positions in their projects, including as project components, as foundation and frameworks for their software, and as tools in their development infrastructure. OSCs appeared to play quite important roles in participants' projects, with some reporting using OSC for key features or foundation in their software or development processes. Some even specifically mentioned OSCs and the open source community as an important or key part of their overall software ecosystem. As for security effects, some participants reported updating their development processes and dependency handling in response to news about vulnerabilities in, or the abandoning of, popular open source projects.

RQ2. *"What are companies' awareness, experiences, and attitudes regarding the security of including external open source code?"* Overall, our participants consisting of software developers, architects, and engineers appeared to be quite aware of the security implications of including OSC in their software, although some reported management not allowing or understanding the concept of open source. Almost all participants reported positive to very positive experiences with open source code, although all except one mentioned experiencing some form of challenge or inconvenience by OSCs in the past, mostly originating from an unmaintained project, a botched patch, or an upstream vulnerability. Our participants seemed to have somewhat ambivalent attitudes about OSC security, with many mentioning that they would or could only handle incidents from OSCs if/when they happen, while their most common security wishes included large-scale audits of their dependencies and OSC projects.

RQ3. *"If and how do stakeholders make decisions and considerations around security and trust challenges of including Open Source Components?"* The decision and selection processes around OSCs reported by our participants appear to span the whole spectrum from purpose-build, in-house components modified by specific teams wrapping and documenting open source projects, to whatever component an individual developer thought right for the job. As for considerations around security, our participants appeared in general to be optimistic, while still acknowledging the large potential attack surfaces of using external code.

Aside from answering our research questions, we discuss some of the broader themes and our interview-spanning findings in greater detail:

Securing a Bowl of Spaghetti. The "chain" part of the software supply chain analogy lends itself to convey an overall image of linear relations, with clear start (producer) and end (consumer) points, with some additional chain links in-between. But in reality, a better fitting picture for the software supply chain in general, and OSS in particular is that of a giant bowl of spaghetti, with many intertwined strands, impossible to discern beginning and ends, even when closer investigating some string. Some companies in our study tackled this problem by focusing only on the security aspects on their plate, namely by maintaining in-house versions or caches of included OSCs, which separates them from many attack vectors in the whole bowl, and allows them to better check and audit the local components. Promising research venues include both the underlying concepts for maintaining such a software stack separation, as well as the necessary tooling like for static analysis, reproducible builds, and package signing. Based on our findings, our recommendations for industry projects include considering established available approaches like version pinning and including static analysis in their build pipeline, as well as to evaluate some of the more recently emerging technologies, like SBOMs and metrics like the OpenSSF Scorecards.

Community of Communities. Our participants seemed to have quite positive attitudes about OSCs, with many mentioning their software or team benefiting from using them, e. g., through reduced maintenance burden, fast iterations, and open communities and code. This exchange can quickly become one-sided, especially as it is not always feasible for both companies to provide, and open source communities to receive, the most common exchange equalizer in the industry: Money. Promising future research opportunities involve identifying ways to best support both individual open source projects in different growth stages and communities, as well as the open source ecosystem as a whole. Based on our current findings, it might be beneficial for companies to approach the open source ecosystem with the mindset of being just another community among the open source communities, instead of treating it as another software supplier. In practice, this could involve the open sourcing of their internal components if feasible, providing guidance and help with issues just as most open source projects, and contributing back if the chance arises.

Not Your Typical Supply Chain. Companies treating the open source ecosystem as any other of their (software) supply chains will likely lead to bad surprises for both sides down the line: Companies might need to scramble if OSCs they had relied on for years are suddenly abandoned by the maintainer or do not implement direly needed features, while open source communities might be punished for their openness by being (mis)treated as a cheaper support desk and alternative for in-house development teams. Unlike a company's other (software) supply chains, the open source ecosystem rarely operates based on contracts, and if a company is not able to provide a value exchange equivalent in money for utilized OSCs, they might want to consider offering some of their developer time or code back to the open source ecosystem. Future researcher venues could involve the legal challenges of the open source ecosystem, best approaches for different company types to support or get involved in open source, and how companies could improve their development processes around involved OSCs. With industry's great power of utilizing freely available OSCs in their software comes also the great responsibility of keeping the open source ecosystem healthy and secure, or as one of our participants formulated it fittingly: "*This is something that we also have in our company policy: Always contribute back.*" (P07)

6. Conclusion

We investigated the use of OSCs in software companies and teams during 25 in-depth, semi-structured interviews with software developers, architects, and engineers. We explored challenges and considerations of software companies and teams around including OSCs in their projects by exploring their behind-the-scene processes, provided guidance and security policies, as well as security challenges encountered in the past and their incident handling. We found that most of our participants' projects had some form of company policy or at least best practices for including external code, with selection and exclusion criteria for OSCs being commonly based on easily visible metrics like activity, number of contributors, or GitHub stars. We also found that most projects contribute in some form back to open source projects, or our participants would at least like to, with some suggesting their management or legal departments do not fully understand the open source ecosystem. Relatively few participants mentioned dedicated security teams or experts, with many wishing for more developer-hours or tools to audit included components, as well as general security checks and pen-tests of their projects and included OSCs.

Acknowledgments

This work is supported in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2092 CASA – 390781972, NSF grant CNS-2206865, and the Google Research Scholar program. Any findings and opinions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. We want to thank all interviewees for their participation and appreciate the industry-insider knowledge and valuable time that they have generously given. We also thank the anonymous reviewers for their valuable feedback.

References

[1] RedHat, *The State of Enterprise Open Source 2020: Enterprise open source use rises, proprietary software declines*, https://www.redhat.com/en/blog/state-enterprise-open-source-2020-enterprise-open-source-use-rises-proprietary-software-declines, Feb. 2020.

[2] W. Turton and K. Mehrotra, *FireEye discovered SolarWinds breach while probing own hack*, https://www.bloomberg.com/news/articles/2020-12-15/fireeye-stumbled-across-solarwinds-breach-while-probing-own-hack, 2020.

[3] C. Cimpanu, *SEC filings: SolarWinds says 18,000 customers were impacted by recent hack*, https://www.zdnet.com/article/sec-filings-solarwinds-says-18000-customers-are-impacted-by-recent-hack/, Dec. 2020.

[4] Check Point Research, *CloudGuard Spectral detects several malicious packages on PyPI – the official software repository for python developers*, https://research.checkpoint.com/2022/cloudguard-spectral-detects-several-malicious-packages-on-pypi-the-official-software-repository-for-python-developers/, Aug. 2022.

[5] L. Tal, *Alert: Peacenotwar module sabotages npm developers in the node-ipc package to protest the invasion of ukraine*, https://snyk.io/blog/peacenotwar-malicious-npm-node-ipc-package-vulnerability/, Mar. 2022.

[6] E. Wittern, P. Suter, and S. Rajagopalan, "A look at the dynamics of the JavaScript package ecosystem," in *Proceedings of the 13th International Conference on Mining Software Repositories (MSR '16)*, 2016, pp. 351–361.

[7] R. Kikas, G. Gousios, M. Dumas, and D. Pfahl, "Structure and evolution of package dependency networks," in *Proceedings of the 14th International Conference on Mining Software Repositories (MSR '17)*, 2017, pp. 102–112.

[8] R. Abdalkareem, O. Nourry, S. Wehaibi, S. Mujahid, and E. Shihab, "Why do developers use trivial packages? an empirical case study on npm," in *Proceedings of the 11th ACM Joint Meeting on Foundations of Software Engineering*, Aug. 2017, pp. 385–395.

[9] A. Decan, T. Mens, and E. Constantinou, "On the impact of security vulnerabilities in the npm package dependency network," in *Proceedings of the 15th international conference on mining software repositories*, 2018, pp. 181–191.

[10] M. Zimmermann, C.-A. Staicu, C. Tenny, and M. Pradel, "Small-world with high risks: A study of security threats in the npm ecosystem," in *Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19)*, 2019, pp. 995–1010.

[11] S. Scalco, R. Paramitha, D.-L. Vu, and F. Massacci, "On the feasibility of detecting injections in malicious npm packages," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–8.

[12] M. Valiev, B. Vasilescu, and J. Herbsleb, "Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem," in *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 644–655.

[13] M. Alfadel, D. E. Costa, and E. Shihab, "Empirical analysis of security vulnerabilities in python packages," in *Proceedings of the 2021 IEEE international conference on software analysis, Evolution and Reengineering (SANER '21)*, 2021, pp. 446–457.

[14] J. Kabbedijk and S. Jansen, "Steering insight: An exploration of the Ruby software ecosystem," in *Software Business*, vol. 80, 2011, pp. 44–55.

[15] D. M. German, B. Adams, and A. E. Hassan, "The evolution of the R software ecosystem," in *Proceedings of the 17th European Conference on Software Maintenance and Reengineering*, 2013, pp. 243–252.

[16] G. Bavota, G. Canfora, M. D. Penta, R. Oliveto, and S. Panichella, "The evolution of project inter-dependencies in a software ecosystem: The case of Apache," in *Proceedings of the 2013 IEEE International Conference on Software Maintenance*, 2013, pp. 280–289.

[17] R. Bloemen, C. Amrit, S. Kuhlmann, and G. Ordóñez–Matamoros, "Gentoo package dependencies over time," in *Proceedings of the 11th Working Conference on Mining Software Repositories (MSR '14)*, 2014, pp. 404–407.

[18] I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci, "Vulnerable open source dependencies: Counting those that matter," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–10.

[19] Y. Wang, B. Chen, K. Huang, *et al.*, "An empirical study of usages, updates and risks of third-party libraries in java projects," in *Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2020, pp. 35–45.

[20] W. Enck, D. Octeau, P. D. McDaniel, and S. Chaudhuri, "A study of Android application security.," in *Proceedings of the 20th USENIX Security Symposium (SEC'11)*, 2011.

[21] I. J. Mojica, B. Adams, M. Nagappan, S. Dienst, T. Berger, and A. E. Hassan, "A large-scale empirical study on software reuse in mobile apps," *IEEE Software*, vol. 31, no. 2, pp. 78–86, 2014.

[22] A. Decan, T. Mens, and P. Grosjean, "An empirical comparison of dependency network evolution in seven software packaging ecosystems," *Empirical Software Engineering*, vol. 24, no. 1, pp. 381–416, 2019.

[23] A. Decan and T. Mens, "What do package dependencies tell us about semantic versioning?" *IEEE Transactions on Software Engineering*, vol. 47, no. 6, pp. 1226–1240, 2021.

[24] Y. Wang, M. Wen, Z. Liu, *et al.*, "Do the dependency conflicts in my project matter?" In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 319–330.

[25] F. R. Cogo, G. A. Oliva, and A. E. Hassan, "An empirical study of dependency downgrades in the npm ecosystem," *IEEE Transactions on Software Engineering*, vol. 47, no. 11, pp. 2457–2470, 2021.

[26] S. E. Ponta, H. Plate, and A. Sabetta, "Detection, assessment and mitigation of vulnerabilities in open source dependencies," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3175–3215, 2020.

[27] C. Liu, S. Chen, L. Fan, B. Chen, Y. Liu, and X. Peng, "Demystifying the vulnerability propagation and its evolution via dependency trees in the npm ecosystem," in *Proceedings of the 44th IEEE/ACM International Conference on Software Engineering (ICSE '22)*, 2022, pp. 672–684.

[28] L. Li, T. F. Bissyande, J. Klein, and Y. Le Traon, "An Investigation into the Use of Common Libraries in Android Apps," in *Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER '16)*, 2016, pp. 403–414.

[29] M. Li, W. Wang, P. Wang, *et al.*, "LibD: Scalable and precise third-party library detection in Android markets," in *Proceedings of the 39th IEEE/ACM International Conference on Software Engineering (ICSE '17)*, 2017, pp. 335–346.

[30] Z. Ma, H. Wang, Y. Guo, and X. Chen, "LibRadar: Fast and accurate detection of third-party libraries in Android apps," in *Proceedings of the 38th ACM International Conference on Software Engineering Companion*, 2016, pp. 653–656.

[31] Y. M. Mileva, V. Dallmeier, and A. Zeller, "Mining API popularity," in *Testing – Practice and Research Techniques*, Springer, 2010, pp. 173–180.

[32] P. T. Nguyen, J. Di Rocco, D. Di Ruscio, and M. Di Penta, "CrossRec: Supporting software developers by recommending third-party libraries," *Journal of Systems and Software*, vol. 161, Mar. 2020.

[33] B. Xu, L. An, F. Thung, F. Khomh, and D. Lo, "Why reinventing the wheels? an empirical study on library reuse and re-implementation," *Empirical Software Engineering*, vol. 25, no. 1, pp. 755–789, Sep. 2019.

[34] F. López de la Mora and S. Nadi, "An empirical study of metric-based comparisons of software libraries," in *Proceedings of the 14th ACM International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '18)*, 2018, pp. 22–31.

[35] ——, "Which library should i use?: A metric-based comparison of software libraries," in *Proceedings of the 40th IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Technologies Results (ICSE-NIER '18)*, 2018, pp. 37–40.

[36] R. G. Kula, D. M. German, A. Ouni, T. Ishio, and K. Inoue, "Do developers update their library dependencies? an empirical study on the impact of security advisories on library migration," *Empirical Software Engineering*, vol. 23, pp. 384–417, 2018.

[37] C. Okafor, T. R. Schorlemmer, S. Torres-Arias, and J. C. Davis, "Sok: Analysis of software supply chain security by establishing secure design properties," in *Proceedings of the 1st ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED '22)*, 2022.

[38] W. Enck and L. Williams, "Top five challenges in software supply chain security: Observations from 30 industry and government organizations," *IEEE Security & Privacy*, vol. 20, no. 2, pp. 96–100, 2022.

[39] M. Ohm, H. Plate, A. Sykosch, and M. Meier, "Backstabber's knife collection: A review of open source software supply chain attacks," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer International Publishing, 2020, pp. 23–43.

[40] N. Nikiforakis, L. Invernizzi, A. Kapravelos, *et al.*, "You are what you include: Large-scale evaluation of remote javascript inclusions," in *Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12)*, 2012, p. 736.

[41] I. Pashchenko, H. Plate, S. E. Ponta, A. Sabetta, and F. Massacci, "Vuln4Real: A methodology for counting actually vulnerable dependencies," *IEEE Transactions on Software Engineering*, vol. 48, no. 5, pp. 1592–1609, 2022.

[42] G. A. A. Prana, A. Sharma, L. K. Shar, *et al.*, "Out of sight, out of mind? how vulnerable dependencies affect open-source projects," *Empirical Software Engineering*, vol. 26, no. 4, p. 59, 2021.

[43] E. Larios Vargas, M. Aniche, C. Treude, M. Bruntink, and G. Gousios, "Selecting third-party libraries: The practitioners' perspective," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, pp. 245–256.

[44] S. Basak, L. Neil, B. Reaves, and L. Williams, "What are the practices for secret management in software artifacts?" In *Proceedings of the 2022 IEEE Secure Development Conference (SecDev'22)*, 2022, pp. 69–76.

[45] N. Zahan, S. Shohan, D. Harris, and L. Williams, "Preprint: Do OpenSSF Scorecard practices contribute to fewer vulnerabilities?" *arXiv preprint arXiv:2210.14884*, 2022.

[46] N. Zahan, P. Kanakiya, B. Hambleton, S. Shohan, and L. Williams, "Preprint: Can the OpenSSF Scorecard be used to measure the security posture of npm and PyPI?" *arXiv preprint arXiv:2208.03412*, 2022.

[47] S. Frey, A. Rashid, P. Anthonysamy, M. Pinto-Albuquerque, and S. A. Naqvi, "The good, the bad and the ugly: A study of security decisions in a cyber-physical systems game," *IEEE Transactions on Software Engineering*, vol. 45, no. 5, pp. 521–536, 2017.

[48] B. Shreeve, J. Hallett, M. Edwards, K. M. Ramokapane, R. Atkins, and A. Rashid, "The best laid plans or lack thereof: Security decision-making of different stakeholder groups," *IEEE Transactions on Software Engineering*, 2020.

[49] A. Poller, L. Kocksch, S. Türpe, F. A. Epp, and K. Kinder-Kurlanda, "Can security become a routine? a study of organizational change in an agile software development group," in *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, 2017, pp. 2489–2503.

[50] D. Baca, M. Boldt, B. Carlsson, and A. Jacobsson, "A novel security-enhanced agile software development process applied in an industrial setting," in *Proceedings of the 10th IEEE International Conference on Availability, Reliability and Security*, 2015, pp. 11–19.

[51] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "You get where you're looking for: The impact of information sources on code security," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P '16)*, 2016, pp. 289–305.

[52] ——, "How internet resources might be helping you develop faster but less securely," *IEEE Security & Privacy*, vol. 15, no. 2, pp. 50–60, 2017.

[53] R. Stevens, D. Votipka, E. M. Redmiles, C. Ahern, P. Sweeney, and M. L. Mazurek, "The battle for New York: A case study of applied digital threat modeling at the enterprise level," in *Proceedings of the 27th USENIX Security Symposium (USENIX Security '18)*, 2018, pp. 621–637.

[54] H. Assal and S. Chiasson, ""Think secure from the beginning" a survey with software developers," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019, pp. 1–13.

[55] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "Sok: Taxonomy of attacks on open-source software supply chains," in *Proceedings of the 44th IEEE Symposium on Security and Privacy (S&P '23)*, pp. 167–184.

[56] S. E. McGregor, P. Charters, T. Holliday, and F. Roesner, "Investigating the computer security practices and needs of journalists," in *Proceedings of the 24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 399–414.

[57] S. E. McGregor, E. A. Watkins, M. N. Al-Ameen, K. Caine, and F. Roesner, "When the weakest link is strong: Secure collaboration in the case of the panama papers," in *Proceedings of the 26th USENIX Security Symposium (USENIX Security 17)*, 2017, pp. 505–522.

[58] C. Chen, N. Dell, and F. Roesner, "Computer security and privacy in the interactions between victim service providers and human trafficking survivors," in *Proceedings of the 28th USENIX Security Symposium (USENIX Security 19)*, 2019, pp. 89–104.

[59] W. Bai, M. Namara, Y. Qian, P. G. Kelley, M. L. Mazurek, and D. Kim, "An inconvenient trust: User attitudes toward security and usability tradeoffs for key-directory encryption systems," in *Proceedings of the 12th Symposium on Usable Privacy and Security (SOUPS '16)*, 2016, pp. 113–130.

[60] K. Gallagher, S. Patil, and N. Memon, "New me: Understanding expert and non-expert perceptions and usage of the tor anonymity network," in *Proceedings of the 13th Symposium on Usable Privacy and Security (SOUPS '17)*, 2017, pp. 385–398.

[61] N. Huaman, B. von Skarczinski, C. Stransky, *et al.*, "A large-scale interview study on information security in and attacks against small and medium-sized enterprises," in *Proceedings of the 30th USENIX Security Symposium (USENIX Security '21)*, 2021, pp. 1235–1252.

[62] M. Silic and A. Back, "Information security and open source dual use security software: Trust paradox," in *Open Source Software: Quality Verification*, 2013, pp. 194–206.

[63] T. W. Thomas, M. Tabassum, B. Chu, and H. Lipford, "Security during application development: An application security expert perspective," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.

[64] L. Bauer, L. F. Cranor, R. W. Reeder, M. K. Reiter, and K. Vaniea, "Real life challenges in access-control management," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2009, pp. 899–908.

[65] R. Barrett, E. Kandogan, P. P. Maglio, E. M. Haber, L. A. Takayama, and M. Prabaker, "Field studies of computer system administrators: Analysis of system management tools and practices," in *Proceedings of the 2004 ACM conference on Computer Supported Cooperative Work*, 2004, pp. 388–395.

[66] D. Botta, R. Werlinger, A. Gagné, *et al.*, "Towards understanding it security professionals and their tools," in *Proceedings of the 3rd Symposium on Usable Privacy and Security (SOUPS '07)*, 2007, pp. 100–111.

[67] J. M. Haney, M. Theofanos, Y. Acar, and S. S. Prettyman, ""We make it a big deal in the company": Security mindsets in organizations that develop cryptographic products," in *Proceedings of 14th Symposium on Usable Privacy and Security (SOUPS '18)*, 2018, pp. 357–373.

[68] F. Jansen, S. Jansen, and F. Hou, "Trustseco: An interview survey into software trust," *arXiv preprint arXiv:2101.06138*, 2021.

[69] J. Ghofrani, P. Heravi, K. A. Babaei, and M. D. Soorati, "Trust challenges in reusing open source software: An interview-based initial study," in *Proceedings of the 26th ACM International Systems and Software Product Line Conference-Volume B*, 2022, pp. 110–116.

[70] M. Gutfleisch, J. H. Klemmer, N. Busch, Y. Acar, M. A. Sasse, and S. Fahl, "How does usable security (not) end up in software products? results from a qualitative interview study," in *Proceedings of the 43rd IEEE Symposium on Security and Privacy (S&P 2022)*, May 2022.

[71] D. Wermke, N. Wöhler, J. H. Klemmer, M. Fourné, Y. Acar, and S. Fahl, "Committed to trust: A qualitative study on security & trust

in open source software projects," in *Proceedings of the 43rd IEEE Symposium on Security and Privacy (S&P 2022)*, May 2022.

[72] M. Fourné, D. Wermke, W. Enck, S. Fahl, and Y. Acar, "It's like flossing your teeth: On the importance and challenges of reproducible builds for software supply chain security," in *Proceedings of the 44th IEEE Symposium on Security and Privacy (S&P 2023)*, May 2023.

[73] K. Charmaz, *Constructing Grounded Theory*. Sage, 2014.

[74] A. Strauss and J. M. Corbin, *Grounded Theory in Practice*. Sage, 1997, p. 288.

[75] J. Corbin and A. Strauss, "Grounded theory research: Procedures, canons and evaluative criteria," *Zeitschrift für Soziologie*, vol. 19, no. 6, pp. 418–427, 1990.

[76] C. Urquhart, *Grounded theory for qualitative research: A practical guide*. Sage, 2012.

[77] M. Birks and J. Mills, *Grounded theory: A practical guide*. Sage, 2015.

[78] N. McDonald, S. Schoenebeck, and A. Forte, "Reliability and interrater reliability in qualitative research: Norms and guidelines for CSCW and HCI practice," *Proc. ACM Hum.-Comput. Interact.*, vol. 3, no. CSCW, Nov. 2019.

[79] E. Kenneally and D. Dittrich, "The Menlo report: Ethical principles guiding information and communication technology research," *SSRN Electronic Journal*, Aug. 2012.

[80] S. Gallagher, *Rage-quit: Coder unpublished 17 lines of javascript and "broke the internet"*, https://arstechnica.com/information-technology/2016/03/rage-quit-coder-unpublished-17-lines-of-javascript-and-broke-the-internet/, Mar. 2016.