Everyone for Themselves?
A Qualitative Study about Individual Security Setups of Open Source Software Contributors

Sabrina Amft*, Sandra Höltervennhoff†, Rebecca Panskus‡, Karola Marky‡ and Sascha Fahl*
*CISPA Helmholtz Center for Information Security, Germany, {sabrina.amft,sascha.fahl}@cispa.de
†Leibniz University Hannover, Germany, hoeltervennhoff@sec.uni-hannover.de
‡Ruhr University Bochum, Germany, {rebecca.panskus,karola.marky}@rub.de

***Abstract*—To increase open-source software supply chain security, protecting the development environment of contributors against attacks is crucial. For example, contributors must protect authentication credentials for software repositories, code-signing keys, and their systems from malware.**

Previous incidents illustrated that open-source contributors struggle with protecting their development environment. In contrast to companies, open-source software projects cannot easily enforce security guidelines for development environments. Instead, contributors' security setups are likely heterogeneous regarding chosen technologies and strategies.

To the best of our knowledge, we perform the first in-depth qualitative investigation of the security of open-source software contributors' individual security setups, their motivation, decision-making, and sentiments, and the potential impact on open-source software supply chain security. Therefore, we conduct 20 semi-structured interviews with a diverse set of experienced contributors to critical open-source software projects.

Overall, we find that contributors have a generally high affinity for security. However, security practices are rarely discussed in the community or enforced by projects. Furthermore, we see a strong influence of social mechanisms, such as trust, respect, or politeness, further impeding the sharing of security knowledge and best practices.

We conclude our work with a discussion of the impact of our findings on open-source software and supply chain security, and make recommendations for the open-source software community.

1. Introduction

Open Source Software (OSS) is a crucial pillar of modern software systems. The popularity of open-source software in companies is at an all-time high and continues to rise [1], [2]. Some popular OSS projects have more than 26 million dependent projects [3].

Given the spread and popularity of OSS, and its integral part in software supply chains, securing the OSS ecosystem is vital for overall software security. Previous OSS-related security incidents, such as the famous Heartbleed [4], log4j [5], or Shellshock [6] vulnerabilities affected millions of projects and billions of users globally [7], [8]. Additionally, some open source contributorss (OSCs) have disastrously used their influence to impact software supply chain security. In 2016, millions of services broke when the left-pad author decided to pull their npm package over a name controversy [9]. Similarly, some OSCs used their software to spread political messages and corrupted it on purpose [10], in some cases even targeting specific populations by, e. g., overwriting local user data [11].

However, incidents are not limited to insecure software, low code quality, or deliberate contributors' actions. They can also occur when the individual security setup of contributors is insufficient, as illustrated by several incidents in recent years.

For example, weak passwords or a lack of multi-factor authentication (MFA) for developer accounts allowed attackers to compromise accounts and abuse them to add backdoors into software or distribute malware [12]–[15]. The list of affected repositories also includes high-profile software, such as ESLint [16] or Gentoo [17].

More sophisticated attacks rely on the inattentiveness of contributors, and enable attackers to gain access based on exposed GitHub API tokens [18] or expired mail server domains, that allowed access to password reset emails [19].

In the above examples, the respective projects have millions of downloads and were abused to distribute malware or collect and steal sensitive data, such as API tokens, passwords, or encryption keys.

These incidents underline the criticality of strong individual security setups of OSCs for software supply chain security, including their Open Source (OS)-related online accounts, physical devices, and sensitive data such as encryption keys and code secrets. However, previous work illustrated that OS can lack guidance, norms, or checklists [20]–[22] related to security, leaving contributors on their own with regard to security best practices, or rely on the potentially sparse communication with other OSCs. In contrast to previous work by Wermke *et al.* [23], who mainly focused on the projects and their internal processes, to the best of our knowledge, our work addresses the individual security measures of OSCs for the first time.

Therefore, we aim to answer the following research questions:

RQ1 *Which technologies and practices do open source contributors deploy for their open-source related individual security setups?*

RQ2 *What are common challenges of securing open source contributors' individual security setups?*

RQ3 *How can open source contributors be better supported in maintaining their individual security setups?*

To answer our research questions, we first searched 100 critical repositories for policies or guidance regarding individual security setups, confirming that projects rarely provide security-related input for contributors. At the core of our work, we conduct 20 semi-structured interviews with OSCs to provide extensive insights into their individual security setups and the measures they deploy to protect their accounts, devices, and sensitive data, as well as their decisions and motivations. While our interviewees were in general security-affine and tech-savvy, they tended to deploy practical and usable measures. Social mechanisms were essential: Contributors rarely talked about their individual security setups to, e. g., not annoy or confront other contributors. However, these mechanisms therefore also imposed obstacles regarding knowledge sharing or necessary security requirements. Based on our results, we provide recommendations to OSS contributors and projects to increase security without impacting the unique culture of trust within OSS.

Replication Availability. To ensure replicability and transparency, we will upload all necessary documents, such as our interview guide, pre-survey and consent form, email templates sent to contributors, and our final codebook in a replication package[1].

2. Related Work

We discuss related research regarding OSS security, previous interview studies, and generic work on expert security advice or measures for personal security setups (personal security setups).

OSS Security Previous work addressed different security aspects in OSS. Most research on OSS security focused on code security or disclosure and on fixing vulnerabilities [24]–[29]. In contrast, we focus on recent works on supply-chain-attacks and OSC security.

In 2022, Ladisa *et al.* systematize attacks on OSS and the supply chain. Using gray and scientific literature, they identified 107 different attack vectors and 33 potential safeguards developers can deploy. They also surveyed 151 experts and developers to validate their findings [30]. In 2019, Zimmermann *et al.* investigated the dependencies of more than 800,000 packages in the npm ecosystem [32]. Overall, they highlighted the potentially disastrous impact single developers may have on the entire ecosystem. Packages, on average, depended on 79 others, and compromised popular ones could impact more than 100,000 other packages [31]. Similarly, Zahan *et al.* analyzed more than 1.6 million npm packages for weak links, i. e., metadata, such as expired developer domains, or the existence of installation scripts. They also surveyed 470 developers about the collected weak links, who confirmed that some, such as expired domains and inactive maintainers, are indeed an issue in the field [33]. Regarding the nature of malicious OS packages, Ohm *et al.* collected and analyzed 174 different malicious code-bases in 2020. Using both compromised benign packages and malicious repositories targeting developers who mistype benign package names, their work depicts the nature of malware on OSS distribution platforms [34]. With the increasing number of attacks on OSS and its developers, platforms, such as GitHub or npm, introduced countermeasures designed to prevent attacks or vulnerabilities. GitHub's security alerts and code scanning were analyzed in 2023 by Fischer *et al.* They discuss the positive impact on code security that both features provided [35].

The presented previous work on supply-chain attacks primarily focused on the characteristics of attacks and malicious code, as well as the impact of compromised software. In contrast, our work aims to shed light on the human factor of supply-chain-attacks and the individual security decisions of OSCs. While previous work identified weak links, we aim to inquire what measures regarding authentication, devices, and sensitive data OSCs take to protect their development environment.

OSS Interviews In the past, various other works conducted interviews to study the unique culture and workflows of the OSS ecosystem.

In 2013, Silic and Back conducted 14 semi-structured interviews with OSCs security professionals and users of security OSS. OSCs often considered themselves as hackers, and argued that many OS security tools are useful for both hacking or security purposes. However, while trusted by security experts, companies were generally reluctant to adopt these security tools [36]. Wen interviewed 13 OSCs in 2018 to investigate knowledge distribution within OS communities. Due to a lack of norms, established rules, and hierarchies, knowledge is only sparsely shared [20]. Similar, Constantino *et al.* interviewed 12 experienced OSCs in 2020 regarding their collaboration behavior and related challenges. Their work highlighted how mainly non-technical issues such as lack of knowledge or documentation can obstruct cooperation between OSCs [21]. In 2023, Constantino *et al.* conducted another interview study with 12 OSCs and a survey with 121 regarding collaboration within the community. Overall, although developers prefer to work alone, collaboration happens and is most common within development and software maintenance [37]. Finally, in 2022 and 2023, Wermke *et al.* published two interview studies within the OSS ecosystem. In the first, they interviewed 27 OSCs regarding background processes within OSS projects, especially related to code security challenges, policies within the project, and their structure regarding commits and releases. They found that OSCs often only react to issues as time is often limited, and that therefore team size has the

1. Available here: https://doi.org/10.17605/OSF.IO/UDT9E

largest effect on measures taken within projects. While their work is closely related to ours, Wermke *et al.* focus on the inner processes of projects and OSCs interactions instead of individual security measures [23]. In a follow-up interview study, Wermke *et al.* interviewed 25 industry developers regarding their use of OSS. Overall, companies usually have guidelines on the inclusion of OSS. While developers would like to support the software they utilize, this is not always possible based on a lack of time or decisions from higher-ups [38].

In contrast to previous interview studies, our work is the first to shed light on personal security setups of OSCs, focusing on the impact and decisions of individuals instead of the projects they contribute to.

Security Advice and Measures Existing policies or personal knowledge can be a decisive factor for the security behavior of developers. In this section, we present prior work on security advice and measures for and from experts.

In 2016, Stobert and Biddle interviewed 15 security experts regarding their password measures. They found that while having higher security standards for important accounts, even experts tended to use the same unsafe measures as non-experts for others [39]. In the same year, Ion *et al.* performed MTurk surveys with 231 experts and 294 end users regarding their security mindset, and find vast discrepancies. Their findings suggest that experts behave more securely, and, e. g., more commonly used MFA or password managers [40]. This work was replicated by Busse *et al.* in 2019, who were able to confirm that their findings still held up [41]. In 2020, Redmiles *et al.* collected 1264 pieces of security behavior and advice from end users, then among others asked 41 experts to evaluate them. They find that while most advice was perceived as reasonable and actionable, experts were in vast disagreements over which advice was the most important and should be prioritized by users [42]. A recent work by Klemmer *et al.* from 2023 asked 18 web developers to search and provide documents containing security advice. After analysis, they found the documents to lack accessibility or consistency, and to be focused on password-based authentication rather than addressing more modern solutions such as MFA or passkeys [43]. Finally, also in 2023, Lykousas and Patsakis crawled public Git repositories in search of leaked passwords to analyze developer password choices. Similar to Stobert and Biddle, they found that OSCs in general have stronger password habits than end users, but again that given a less critical context, they make similar weak password choices [44].

Previous work on security-related advice and measures has mostly focused on a generic sample of security experts or developers. In contrast, we aim to identify which measures OSCs apply in practice, and what unique challenges they face in the context of contributing to OSS.

3. Methodology

Below, we provide an overview of the methodology of our interview study, including details on our semi-structured interviews regarding design, recruitment, and coding. We further discuss the ethics and limitations of our work.

3.1. Interview Structure

While previous work was primarily interested in open-source code security or disclosure of security issues, we focus on identifying and understanding individual security setups and decisions of OSCs. To gain in-depth insights into the status quo, we conducted semi-structured interviews with OSCs.

As a first step to designing our interview guide, we used our dependency-based recruitment list (cf. Section 3.2) to identify the 100 projects with the most dependents on GitHub for the presence of security policies and whether they provide contributors guidelines regarding their individual security setups. However, we could not find any meaningful guide beyond providing generic contribution guidelines or commonly naming contact information for responsible vulnerability disclosure [45]. We, therefore, omit a detailed analysis but use this as a further motivation to identify common practices and challenges and provide recommendations and guidance for both the open source and security research community. We based the initial draft of our interview guide on our research questions, and then expanded it based on related work and past incidents related to incidents in the OSS ecosystem. We conducted two initial pilot interviews, to improve the interview guide and performed only minor adjustments during our study in the form of, e. g., additional nudges, or question rephrasings.

For ease of reporting, we divided our interview guide into seven sections, excluding the introduction and debriefing. Figure 1 and the following give an overview of our interview guide. The full guide can be found in our replication package (see 1).

Before the interview itself, we gave a brief introduction about our institutions and our research. We further stressed that answers were voluntary and that interviewees could leave at any time, and gathered their explicit verbal consent to our audio recording. Additionally, interviewees were offered to conduct the interview in German instead of English where applicable if they were more comfortable with it.

1. Project Demographics: To break the ice and gain some insights into the interviewees' contributions for our later analysis, we asked about their projects, including questions regarding their purpose, but also about the length and type of the interviewees' involvement. Additionally, we were interested in financial supporters and users of the project.

2. Individual Security, Privacy & Data Safety Setup: After gaining an initial overview of their work, we asked interviewees about their individual security-, privacy- and data safety-related setup. We stressed that we were only interested in measures that affected their open source work, i. e., that we were not interested in measures they solely used for, e. g., private accounts, or their day job. This section included questions about accounts and devices used for their open-source contributions, and related security measures

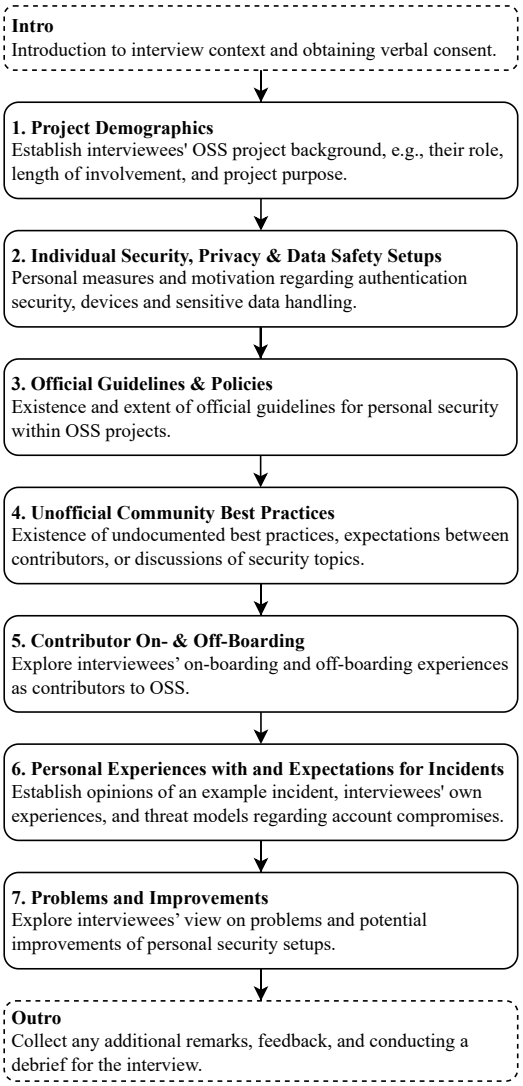

Figure 1. Overview of the interview structure and the topics we addressed. We asked interviewees broader questions, with more detailed follow-ups in each section. Depending on the dynamic of the interview, deviations from this structure were possible.

such as authentication, encryption, or storage measures. Furthermore, we asked how sensitive data such as code secrets, API, SSH, or encryption keys were handled or shared. We also asked about their primary motivation for their setup and changes over time or differences to their private security setup.

3. Official Guidelines & Policies: Companies often provide official and binding policies regarding security and privacy for employees. While OSC are typically volunteers, their individual setup can affect millions of users after successful attacks, therefore, we were interested in whether OSC encountered any kind of official security policy within the projects they contributed to. If present, we asked several follow-up questions about their specifics, e. g., what they covered and how they were communicated or enforced. In case there were no official policies, we asked if there was a specific reason for their absence.

4. Unofficial Community Best Practices: Based on previous work on the social interactions of OSS contributors [23], [36], we expected our interviewees to share or discuss actively, e. g., news from the community or general technology, including incidents and security measures. We therefore differentiated between the official guidelines from the previous section and also included several questions related to community discussions, security expectations for specific people and roles, and overall incentives.

5. Contributor On- & Off-Boarding: Similar to the third block in our interview guide, this block was inspired by workflows within development teams in the industry that typically do not apply for OSS projects. We were therefore interested in the degree to which OSC encountered on-boarding instructions regarding expected security behavior, or how off-boarding is handled when somebody with elevated rights decided to leave a project.

6. Personal Experiences with and Expectations for Incidents: This section was designed to capture interviewees' opinions on past incidents, challenges they have encountered with their setup related to security, and any inconveniences they have experienced within their projects. We asked them for their opinion of the ctx-compromise in 2022 [19], providing details when necessary. We further asked how severe they gauged their risk for similar attacks, i. e., their personal threat model. Additionally, we inquired about incidents they experienced themselves or heard of.

7. Problems and Improvements: To conclude the interview, we explored interviewees' perspectives on the overall problems within security setups in OSS, improvements they would love to make if they had no limitations, and lastly, the security measures they deemed most important.

Finally, if there were no other remarks or questions, we concluded the recording and handled compensation details. We gave them further opportunities to ask questions and offered to store their email address to receive a pre-print, to provide them with the opportunity to veto phrasings to ensure we did not misinterpret them or reveal too much about their identity.

3.2. Recruitment

To collect an initial list of critical open-source projects, we used GitHub Search [46], [47]. GitHub Search is a continuously updated sample of GitHub repositories with at least ten stars that are written in one of the 20 most popular programming languages on GitHub[2]. To only include active open-source projects, we excluded projects with less than 40 commits and less than 20 different contributors overall, as well as projects that did not receive any commits within the last six months at the time of collection [23]. Our final sample included 44,468 projects.

We sorted this list by highest dependent count according to GitHub to identify the projects with the broadest outreach

2. This list can be seen on their website: https://seart-ghs.si.usi.ch/.

for the first half of recruitment. These dependencies stem from references within code on GitHub, i. e., if a project is called upon by another, its dependency count is incremented. However, as many relevant projects are typically directly installed and not built upon by others, they were not reflected in our dependency-based list. We, therefore, created a second list of repositories, this time sorted using the popularity score proposed by Wermke et al. [23]. We refrained from sorting purely based on stars or forks, as we noticed high false-positive rates (e.g., non-code repositories) within both. We followed both lists from top to bottom, equally visiting each repository to recruit interviewees. We searched the projects and contributors for public websites and contact information external to GitHub, acknowledging the GitHub ToS regarding personal data usage [48]. We refrained from emailing anybody more than once and did not contact individuals who either did not provide public contact information or explicitly stated they did not want to be contacted for advertisements, job offers, or research studies.

We approached our interviewees with an email including links to our project and team website for further information regarding our work and our Qualtrics pre-survey [49]. The latter was used as a consent form to collect demographic information regarding the interviewee's OSS projects and experience. We forwarded them to our Calendly [50] to schedule a time slot for the remote interviews. We further offered interviewees to choose their preferred video conference tool between institution instances of Zoom, BigBlueButton, or our self-hosted Jitsi.

As compensation for an estimated hour of their time, we offered each interviewee to sponsor a GitHub project of their choice with $60.

Overall, we recruited 20 interviewees until we reached thematic saturation. We additionally aimed to recruit a diverse set of interviewees based on their project type, size and outreach, as well as the contributors experience, including their years of experience and project roles. Table 1 overviews our interviewees, the projects we recruited them through, and additional recruitment information.

3.3. Data Analysis

We used a semi-open coding approach based on thematic analysis [51]. Before the first codings, we developed an initial codebook using our expectations for common themes based on our interview guide and related work (see 1. We further enriched this codebook using our two pilot interviews. Although active OSCs, these pilots did not satisfy our recruitment criteria and were therefore not included in our results. To increase the reliability of our results, each transcript was coded by two researchers individually using Atlas.TI [52]. Since we conducted interviews in English and German, all involved researchers were fluent in both to ensure a correct understanding of the transcripts.

After coding, the researchers merged and discussed their codings until all conflicts were resolved. Prior interviews were revisited whenever a new code emerged in this process to ensure consistent coding.

We assigned 1,581 codes over all interviews, averaging on 79 codes per interview.

In line with previous work, we chose to omit any form of inter-rater reliability [23], [53]–[55] in favor of a coding approach based on regular discussion and result merging.

3.4. Ethics

This work was approved by our institution's Ethical Review Board (ERB). Additionally, we modeled our study on the ethical principles for research involving information and communication technologies presented in the Menlo Report [56].

During recruitment, we relied on cold emailing. However, we contacted only individuals with public-facing contact details and sent no additional invites or reminders. All interviewees signed a consent form including details on our interview procedure and data handling before signing up for the study. We further asked for permission to record the interview audio. During the interview, we reminded them about some core parts of the consent form, such as their option to skip any question and leave and revoke their consent at any time, in which case we would delete all data stored about them. While we collected some identifiable information, such as their name and email, this was used to send interviewees a pre-print of our work to allow them to correct misunderstandings. We handled and stored all collected data in compliance with the EU's GDPR. All data was stored in a self-hosted encrypted cloud, and transcriptions were done using the GDPR-compliant service Amberscript and manually checked for correctness. We removed all audio recordings and transcripts after paper acceptance. For their participation, we offered interviewees to sponsor a GitHub project of their choice with a one-time sponsorship of $60, in line with previous research [23].

3.5. Limitations

For our work, typical interview limitations apply, including common biases such as under- or overreporting, selection, self-reporting biases, and recall and social desirability biases. This includes that OSCs with an especially (in)secure behavior might have chosen to ignore our invitation as they either felt like they could not contribute due to their lack of security or that we are not trustworthy, and that telling us about their measures would be contrary to their threat model. We sampled interviewees from projects with either a high number of dependents or a high popularity. We did not regard the interviewees' recent project activity or extent of contribution besides requiring them to have made at least 10 commits, and therefore, in some cases, we talked to developers whose contribution was only minor or who had moved on for years. However, we argue that this type of involvement is also part of the OSS ecosystem and that previous incidents have shown that, e. g., project abandonment and outdated security decisions could also harm security.

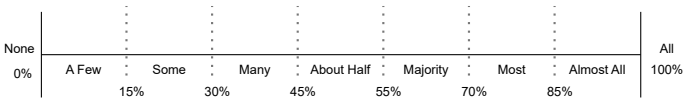

Figure 2. Overview of the qualifiers and respective percentages used throughout our results. All occurrences of the respective qualifiers always refer to the same portion of participants.

Overall, our work is qualitative and should be interpreted in context. Based on the present biases and sampling choices, it does not generalize to the overall OSC population. Finally, we collected our list of initial repositories in July 2023 but invited interviewees and conducted interviews until November 2023. Due to this time, the activity of some projects and OSC might have shifted.

4. Results

Below, we report on the results and main findings of our semi-structured interviews. While roughly following our interview guide, we omitted or merged some less engaging or impactful interview sections for brevity. We will provide more detailed results and our complete codebook in a later replication package. In our results, we generally avoid reporting counts due to the qualitative nature of our interview study. To avoid a quantitative interpretation of our work, we will mostly rely on qualifiers (see Figure 2) to give a general idea about the prevalence and weight of our findings.

4.1. Interviewee Demographics

We report interviewee demographics and their OSS involvement. Table 1 summarizes important demographic information of our interviewees. We did not collect the gender, age, or ethnicity for lack of relevance to our research questions and to protect our interviewees' privacy.

Almost all had on average more than six years of experience doing OSS, and 11 spent five hours or more of their weekly time on OSS tasks. Nine worked on OSS in their free time or as a hobby, while eleven were paid by their company to spend at least a fraction of their weekly working hours on OSS. Besides companies paying developers to work on OSS, twelve received donations, of which nine stemmed from company sponsorships. In contrast, four were employed by non-profit organizations or governments. Seven mentioned receiving donations from individuals. In three cases, their work was financed by other products of the company they worked for or support contracts.

We asked interviewees to describe their role in OSS projects. As interviewees typically were active in multiple projects, many filled different roles simultaneously or had varying degrees of involvement. Six reported to own at least one project themselves. Overall, twelve were active maintainers in at least one project, and six were the sole contributor. Six also reported a less active role, and mentioned only infrequent or triaging commitment. Regarding their typical team size, three worked in small teams of less than five contributors, while five mentioned larger groups. We further inquired about the platforms they used to communicate and work on OSS together. All interviewees mentioned regular usage of GitHub, and some also used other social coding platforms such as GitLab or BitBucket. Nine also had accounts with package managers for their respective primary programming language, e. g., npm for JavaScript developers, or PyPI for Python users. Regarding communication, eleven of our interviewees regularly used a messenger, typically Discord or Slack, to chat with other project members. However, interviewees also commonly reported interacting via GitHub issues, especially when communicating with contributors beyond the core team or external developers: "*For the people who are official maintainers, we discuss things on Discord [...]. For other contributors, we mainly discuss things via the GitHub interface [...].*" (P16).

Overall, eleven reported they still were involved in the project we recruited them through. Regarding their overall projects, seven contributed to code (re)structuring projects, such as linters, validators, or parsers. Other common project types included web design libraries and tools, or application frameworks. Sixteen mentioned companies using their software, eleven including well-known platforms for, e. g., social media, entertainment, or news outlets.

Overall, the interviews lasted between 34–85 minutes, averaging 56 minutes. We conducted 14 interviews in English, and six in German.

> Summary: Project Demographics.Interviewees were generally highly experienced, and half could spend at least a portion of their full-time employment on OSS. Most were active contributors in at least one project; some only had a less active triage role. Most common project areas included code (re)structuring tools, web design standards and libraries, or application frameworks.

4.2. Individual Security Setups

After inquiring our interviewees about their projects, we asked about their personal security setup, i. e., all measures they took to protect their accounts, devices, and sensitive data related to their OSS work. Overall, most interviewees were security-affine and aware of risks. Almost all reported the use of both MFA via timed one-time password apps or hardware tokens, and a similar portion mentioned their password managers. Many interviewees further reported the adoption of MFA where possible, but "*Not every account supports YubiKeys or 2FA.*" (P1).

However, interviewees also discussed obstacles by service-specific policies restricting, e. g., password length or not offering certain MFA types. "*Websites tend to have rather odd, and usually not considered secure in my view, requirements that narrow down the list of possibilities.*" (P14).

Despite upcoming hopes of establishing passkeys [57] as a novel authentication technology, only a few interviewees mentioned their use.

Besides secure authentication measures, we asked interviewees about the device they used to work on OSS. Overall,

TABLE 1. DETAILED OVERVIEW OF INTERVIEW PARTICIPANTS, THEIR PROJECT BACKGROUND, AS WELL AS SOME PROJECT METADATA. FOR REPORTING, PARTICIPANTS WERE ASSIGNED AN ALIAS. WE ONLY REPORT BINNED PROJECT METRICS TO PRESERVE BOTH OUR PARTICIPANTS' AND THEIR PROJECTS' PRIVACY.

Alias	Interview			Recruitment Project			Self-Reported OS Experience[2]		
	Lang.	Duration	Codes[1]	Group	Type	Dependents	Role	Years in OS	Weekly Hours
P1	GER	55:06	90	Dependents	Transpiler	> 10 Mio.	Owner	6-10	1-5h
P2	EN	53:22	98	Dependents	Static Code Analyzer	> 10 Mio.	Infrequent	>10	1-5h
P3	EN	1:09:48	67	Dependents	JSON Validator	> 10 Mio.	Maintainer	6-10	1-5h
P4	EN	50:32	59	Dependents	CSS Optimizer	> 10 Mio.	Regular	3-5	1-5h
P5	EN	42:09	67	Dependents	JavaScript Optimizer	> 10 Mio.	Owner	6-10	1-5h
P6	GER	38:46	67	Dependents	Web Resources	> 10 Mio.	Maintainer	>10	>40h
P7	EN	58:50	58	Dependents	JavaScript Parser	> 3 Mio.	Maintainer	6-10	Unsure
P8	GER	55:07	89	Dependents	Type Checker	> 5 Mio.	Maintainer	6-10	>40h
P9	EN	58:54	122	Dependents	Logger	> 5 Mio.	Owner	>10	11-20h
P10	EN	1:24:58	83	Score	Data Visualization	< 1 Mio.	Maintainer	>10	1-5h
P11	EN	56:27	86	Score	App. Framework	< 1 Mio.	Infrequent	3-5	1-5h
P12	EN	1:00:01	80	Dependents	Unit Testing	> 5 Mio.	Maintainer	6-10	>40h
P13	GER	1:12:19	60	Score	App. Framework	< 1 Mio.	Owner	>10	>40h
P14	EN	1:04:09	117	Score	JavaScript Library	< 1 Mio.	Maintainer	>10	>40h
P15	EN	57:41	65	Dependents	CSS Optimizer	> 10 Mio.	Owner	>10	Unsure
P16	EN	57:25	85	Score	Shell Configuration	< 1 Mio.	Maintainer	6-10	1-5h
P17	GER	48:11	77	Score	App. Framework	< 1 Mio.	Maintainer	6-10	21-40h
P18	GER	33:29	58	Score	System Utilities	< 1 Mio.	Maintainer	3-5	1-5h
P19	EN	52:10	76	Score	Static Site Generator	< 1 Mio.	Owner	6-10	1-5h
P20	EN	57:58	69	Score	Version Control	< 1 Mio.	Maintainer	>10	1-5h

[1] Total number of codes assigned to the interview after resolving conflicts. [2] According to pre-survey

almost all used a laptop for OSS, and only some mentioned a desktop computer. While half of our interviewees used mobile devices such as smartphones, they were typically not used to write source code but for communication and MFA apps. About half used multiple devices. Half of all interviewees mentioned securing these devices using biometric authentication in addition to their passwords.

Our interviewees typically avoided lesser usable or effective measures instead of applying a wide variety of security measures, regarding which P14 describes that the community has "*a few paranoid people, and then there's everybody else*".

For example, while most mentioned using device encryption at rest, about half relied on the default encryption built into their operating system. Some seemed unsure whether they genuinely had encrypted systems. Related, only about half mentioned to sign their commits regularly cryptographically. Some described commit signatures as hard to use or an ineffective security measure:

> "*But I don't do [commit signing] on every project because sometimes it's too much work. [...] I'm not sure if it functionally has a bigger advantage. As long as my GitHub account is already pretty secure.*" - P19

Physical security was rarely perceived as having high relevance. About half described their home as secure enough without additional measures, especially since some mentioned working from home, which meant that their devices also stayed there. "*I just keep them at the house. It's really secure there. [...] I'm doing home office so they're just on the desk.*" (P4).

While traveling, about half mentioned measures such as not leaving their devices unattended. Some additionally mentioned situational measures such as not leaving the devices in a parked car or carrying them separately from the physical keys required to unlock these devices. "*I never leave my laptop unlocked and also just always with me. I never leave it in a vehicle; I live in San Francisco.*" (P12).

Overall, the digital security of our interviewees seemed more robust and more thought-through than their physical security, which might be due to the perceived severity of physical attacks. While the overall likelihood of attacks was perceived as low, they were worried about attacks explicitly targeting them rather than just petty theft. Some, therefore, expected attackers to be highly dedicated to gaining access to their unlocked devices, to a degree where weaponized violence against them was a reasonable possibility.

> "*If somebody breaks in my flat and is inside, they can torture me to unlock my devices. [...] But if somebody actually targets me and gets physical access, there's no realistic way to defend against that*" - P8

In other words, when physical security becomes relevant, the situation is dire enough that it would not help and is therefore not a sensible security measure: "*Physical access will almost certainly undo most security attempts. If I encounter a threat actor who gets physical access to my devices, I'm probably screwed.*" (P3).

Beyond physical security for devices, many interviewees were cautious with their network and device structure, using, e. g., private networks or multiple virtual environments to keep devices and especially projects separate from each other.

As a final topic regarding their individual security setups, we asked interviewees which sensitive data they might need to handle. We prompted for code secrets, such as API

keys, or personal secrets, such as SSH or encryption keys. Almost all mentioned SSH keys and many stated to handle encryption keys. Code secrets were only relevant in some cases. We attribute this to how secrets in some projects are automatically handled, e. g., by using private repositories that only authorized project members had access to:

> "*When I make a repository for a new piece of code, I go to [our admin] and tell him which secrets I need in my build scripts. He unlocks them for me and I only use variables. I don't know the secrets, I don't know how often they're rotated.*" - P13

About half interviewees mentioned how secrets were limited to higher access within a clear hierarchy, i. e., since only specific project members were allowed to handle, e. g., release tokens for npm, there was no need to share these secrets in any way. Some described how they were generally cautious around various web resources. For example, a few mentioned carefully vetting or auditing browser plugins, while many reported they consciously chose to use their respective browser. About half used plugins to block advertisements or trackers. Although many interviewees mentioned using separate browsers, profiles, or even GitHub accounts to distinguish between (OSS) work and private usage, the majority stated that overall, their security setup for private or OSS environments was primarily the same, further stressing how our interviewees were generally tech-savvy and security-aware even for personal purposes.

We were further interested in our interviewees' motivations, and influences regarding their security setups. Many interviewees mentioned a personal responsibility for their code and users:

> "*[...] I realized the security risk that having my package on lots of machines could cause. I am aware that I am responsible for what is pushed, and what is pushed could be bad, right?*" - P5

Similarly, hearing about or experiencing incidents was a motivation for about half to improve their individual security setups as well. For example, P7, whose repository used by millions was hijacked based on developer credentials, commented: "*The incident became an eye-opener for me. Before [it], I wasn't much aware of security, because I am not involved in... I just want to code.*" (P7).

Beyond the negative impact of security issues, many interviewees further reported news articles and publications that influenced or inspired them to upgrade their security. "*I'd read some article about how to sign your git commits and all this stuff, I was like, "Oh, that seems useful." I started doing that.*" (P9).

Some others were incited by what they heard or noticed others do, as P11 describes: "*I see most people do that, so I also do that.*".

Finally, following their general tech-savvyness, many reported being personally interested in security topics and, therefore, enjoyed testing or exploring security measures.

> Summary: Individual Security Setups.Almost all interviews used MFA and password managers to increase account security. Beyond this, measures become rarer, but include device encryption, code signing, the separation of networks and devices, or using continuous integration to handle code secrets. Physical security played only a secondary role. Overall, interviewees had a solid security-awareness and tech-affinity, and tended to only deploy measures they perceived as effective or practical.

4.3. (Un)Official Guidelines

After learning about the security measures interviewees deployed and about their motivations, we were interested in security measures-related instructions or discussions within OSS projects.

First, as suspected after our initial motivational search of written guidelines in the 100 repositories with most dependents, no interviewees reported any official security guideline in their projects besides generic contribution regulations or policies for vulnerability disclosure. Despite that, a few felt that personal security guidelines within projects had some merit. Beyond projects, some interviewees described regulations that originated either from their workplace or organization and that indirectly affected their OSS projects or were still adhered to, although the project was nowadays independent of their original company.

> "*All of these [...] were governed by companies anyway in the first place. Companies themselves have security policies. I think they assume that just goes down there. They don't usually grant maintenance access to random people*" - P2

One interviewee, who is active in a widely used application framework, described these company measures as too pervasive, as their OSS team did not typically utilize the regular company workflows:

> "*The open source team is actually on a war footing with the topic of managed computers. Because that means antivirus was installed by the company, and we don't know anymore what's running on our devices, we have no control. [...] We have coworkers who prefer buying their own device for work, because they don't need [the managed software and secrets].*" - P13

Some others mentioned how projects enforced the use of MFA for their GitHub accounts if they were part of specific projects or organizations, a measure that GitHub has been gradually rolling out and plans to extend to every account by the end of 2023 [58]. Interviewees had various explanations when asked why no official guidelines were provided by OSS projects. First, a majority felt that official guidelines lacked relevance. This included interviewees not having the time to write guides, that other tasks had higher priority, or that a project was not security-critical, and therefore, no harm could be done: "*Most of the ones I've worked on aren't dealing with a whole lot of sensitive data.*" (P9).

Furthermore, the individuals' security was not perceived as important enough to enforce or request security measures that might deter other contributors. About half of our interviewees mentioned that they either implicitly trusted other contributors and their security setups or that the team was so small there was nobody to write security guidance

for: "*Because I'm the person who presses merge on pull requests and I'm the person that deploys, I don't need to communicate the guidelines to anyone else.*" (P5).

In this case, the respective project had hundreds of contributors, and more than ten million dependents. In some other cases, interviewees assumed a lack of responsibility, in which they did not feel up to the task of deciding which measures are practical and sensible or perceived the specifics, such as the exact password manager users should install, as up to the individuals and not part of the project.

> "*Part of me is thinking what it would take to document that. I think part of it is that it's not project-specific usually. [...] People might use different tools to accomplish the same thing.*" - P14

One interviewee further mentioned that liability could be an issue if official regulations were published and were insufficient. Finally, some interviewees argued that the default security measures, such as GitHub requiring MFA, were plenty and therefore saw no need to implement other measures.

> "*I would need to communicate these guidelines and ask for people to do that but even if that were the case, I think npm and GitHub would be enough to enforce these. I could change the settings of the organization to enforce [2FA].*" - P5

Regarding less official communication and floating knowledge, interviewees rarely described other methods from contributors than what they already applied. We attribute this to how contributors trust that each other's setups are secure: "*I think it's just mental fallacies that you're like, "Hey, he's a friend. I can trust him.", and then he turns out to be a very terrible roommate.*" (P2).

Similarly, contributors do not want to pressure or anger others by communicating expectations:

> "*You can't enforce certain things. I've got friends that store and reuse the same 5 passwords in a word document. And I can't eliminate that possibility within open source. Like, I don't want to ask my colleagues, do you use this or that?*" - P1

Overall, most of our interviewees stated that they never or only rarely discussed personal security measures with other OSCs. In some cases, security was discussed, but it mainly encompassed tooling choices or notifying each other about potentially lacking security: "*In the member lists, you can see who activated 2FA, and people tell you it's important to have if you're part of certain organizations.*" (P6).

We further inquired interviewees about expectations towards their and their team's security behavior, which were mentioned by a majority. We found that similar to how OSCs rarely discuss security with each other, expectations are also typically not directly communicated and implicit. However, many mentioned different security measures—mostly usage of MFA and solid password hygiene—as an expectation between contributors: "*If I learned a contributor is not using 2FA on their account, I would be shocked*" (P2).

Additionally, about half of interviewees mention access rights or hierarchies within projects. This included how often only a few contributors can release a package and how others can, at best, submit pull requests but not directly commit to a project. These mentions often included the implicit expectation that the respective project members surely also had a higher security level, as their impact in case of compromise was higher: "*For people who are just outside contributors, mostly we don't expect any security. [...] If it's like an admin account, then, of course, we need top-level security.*" (P11).

> Summary: (In)Official Guidelines.No OSS project provided guidelines for individual security setups. Exceptions were projects with organizational roots or relations, where company policies also affected OSCs. Reasons given for this lack of official guidance included a lack of relevance or responsibility for others' security, and trust towards the other contributors that made policies superfluous. Regarding unofficial best practices, we only rarely found interviewees to discuss security with each other, and found that they typically by default expected others to have a sensible security setup.

4.4. Contributor On- & Off-boarding

Next to the guidance and regulations interviewees encountered, we also inquired about their experiences with on- or off-boarding in OSS projects. Only a few encountered policies for their individual security setup during on-boarding. We attribute this to the general lack of guidelines within OSS projects. Similar to what interviewees told us about overall security guidelines, on-boarding measures also often happened in the context of company policies or enforced security measures by, e. g., GitHub. Overall, off-boarding measures were described by most interviewees, and typically encompassed access restrictions.

Some interviewees mentioned that they gained project access rights or even ownership overnight. For example, P1 commented about a compiler project with millions of dependents:

> "*For <project> I wrote the original creator a message, and he told me he'll give me access, and then I got an email that I'm now the owner of <project>. That was a surprise for me, I didn't know him, but he trusted me a lot.*" - P1

These decisions were typically made based on reputation gained via regular commitment or since the original owner required an inheritor: "*[They] got burned out or just moved on [...] I was one of the last to file issues or do a patch and it was like, oh, here you go. You're now in the repo.*" (P10).

In contrast to company employments, where contact expiration marks the end of an individual's involvement and triggers off-boarding and access restrictions, there is no similar event for OSS volunteers. On the contrary, about half told us about OSCs who stopped contributing but never officially lost their access rights. This was explained by the circumstance that, due to the voluntary nature of the work, determining what constitutes a person leaving and what is just a lengthy break to prioritize other areas of life is a nontrivial task. "*I think people just get busy. I know for*

some of my projects I've gone a year between commits. I didn't stop, I just had other stuff." (P4).

Others perceived removing others' access without their confirmation and approval as overstepping their bounds, although their projects had millions of users, implying that incidents might have vast consequences. "*If somebody hasn't been active for three years, like do I revoke their access? It seems like I'm kicking them out.*" (P14).

> Summary: Contributor On- & Off-boarding.Interviewees rarely encountered official policies during on- or off-boarding, especially outside of company contexts. Off-boarding encompassed access removals. Within OSS projects, contributors could get access relatively easy, while they tended to keep this access even when they eventually became inactive.

4.5. Challenges, Improvements, and Recommendations

In the following, we report on the challenges that OSS faces regarding personal security practices, improvements interviewees would like to make on their individual security setup, and measures they recommend the most to achieve solid baseline security.

OSS Challenges Regarding Personal Security About half of our interviewees criticized dependencies in OSS, especially regarding JavaScript projects. Common issues, especially mentioned by participants active in popular JavaScript projects, included too many dependencies and large quantities of unknown code that could not feasibly be audited, leading to security issues:

> "*They use npm, and there's tens of thousands of packages being pulled in by as many different people who were never vetted. Who were never asked, consulted, or reviewed in any way.*" - P14

Some challenges interviewees perceived in the security behavior of OSCs were attributed to how security was (not) communicated. This included how some did not know the setup of others. These interviewees also argued that security could not be enforced and that there were unrealistic expectations towards OSCs to be secure and deliver quality.

> "*There's no centralization. Everyone's going to do their own thing. [...] If only one out of a thousand developers have very bad security practices, [incidents] are going to happen, because there are thousands and thousands of open-source projects.*" - P5

While some mentioned the unique trust structures as a further issue on how security behavior lacked transparency, this was typically seen positively or as a less severe issue: "*There's no visibility into the chain of trust. There's no solid legit trust, it's just always implicit, which has worked so far.*" (P2).

Despite this sense of security, a major and widely used tooling repository that the interviewee had access to, was hijacked due to weak developer credentials. As they had not been an active contributor when the hack occurred, we assume they were unaware of this incident. Many interviewees also described human factors as issues, such as owners abandoning projects, that security might lack if it is not enforced, or that projects security can be lacking due to unaware or inexperienced developers. "*There's a lot of really young developers who are not so aware of all the possibilities and the creativity of security malicious people*" (P4).

Furthermore, many mentioned technical issues, such as a lack of usability of MFA or code signing, the use of local builds, lack of isolation between projects, or a lack of version pinning regarding safe dependency versions. While the latter was mentioned as a remedy to the issues with unvetted dependency updates, interviewees felt that this option was not widespread enough: "*[Without it] you will be installing random versions, and then you'll be vulnerable to these attacks [...] Dependency pinning with lock files must be the default across the industry.*" (P2).

Finally, many interviewees mentioned that individual security is often a secondary or less critical task: "*I guess there's still some naivety or laziness. It's not the most exciting topic and feels more like an obligation.*" (P6).

Overall, the creation of OSS is the main goal, and security is seen as a task that takes away from contributors' available time, especially if they are unable to pay their bills with their OSS work:

> "*I think a lot of people who maintain open source projects, [...] probably have some other form of employment, that is likely a full-time commitment, [...]they don't have a lot of time to really maintain it. That in turn means that vetting of external contributors or doing security audits of open source libraries is always going to be hard.*" - P3

Improvements of Interviewee Setup During our interview, we spoke about incidents and how interviewees rated their setup, and perceived themselves at risk of an attack. We found that a majority perceived their risk as low, with a few others being unsure about it. Typical reasons included that they had no critical access, were on top of their security, or that secure defaults such as enabling MFA were sufficient. "*I would say very low because, again, everything is behind 2FA. I'm very on top of my emails and domains.*" (P2).

However, about half of our interviewees mentioned that perfect security did not exist because there was always something unexpected, such as code vulnerabilities circumventing security measures or a physical intruder, that might break through the measures they considered secure. "*Some completely unknown zero-day vulnerabilities could be exploited in anything that I use. Generally, I feel my setup is probably better than the average of most people.*" (P19).

About half of the interviewees wished to improve their authentication security, e. g., by moving to a different password manager or by upgrading MFA to a hardware key.

Another half further mentioned that they would like to become more careful, e. g. by auditing their tools or separating their devices regarding their purpose, including the adoption of sandboxing and virtual machines:

> "*I would run everything in VMs and move applications into flatpaks. I'd have way more security boundaries on my PC. Not only the devices, but also separating applications and reproducers. Just everything imaginable.*" - P17

Similarly, some mentioned technical issues such as malfunctioning hardware keys, or issues with the sharing or syncing of password managers obstructing them. Interviewees argued that resolving these problems would also improve their setup. "*There should be better tools. Then it, you know, that makes it easier to demand that as a requirement of a contribution.*" (P9).

Regarding encryption, some interviewees mentioned deploying encryption for both their devices or email, which they did not (yet) adopt due to bad usability: "*I already thought about encrypting email [...]. I had that a while ago, but then something didn't work, and I didn't bother to get it running again.*" (P18).

Some interviewees would appreciate more time to further educate themselves besides mentioning specific security measures: "*Maybe actually spend some time investigating how to make [my setup] better [...] there are lots of resources on the internet on how to make it more secure.*" (P15).

This also included interviewees who mentioned that checklists with instructions beyond pure education would help them improve their security: "*It would be helpful if there were checklists or similar, that you could follow, especially regarding two-factor authentication and commit signing.*" (P6).

Finally, some interviewees were already satisfied with their individual security setup and therefore did not want to upgrade their setup.

Interviewee Security Recommendations Finally, we asked interviewees what security measures they considered a bare minimum that all OSS contributors should adopt. Overall, interviewees suggested the measures they applied, underlining that they were content with their measures.

By far the most common, almost all recommended using MFA, which was often considered the most critical measure that prevented account compromises. "*With 2FA, you can be quite sure that even if they know your password, they won't be able to do anything. Please at least do that.*" (P4).

Furthermore, about half interviewees recommended using a password manager: "*I always tell people I know who are not using a password manager to use one. That's one of the most basic things people can do [...]*" (P19).

Besides password managers, about half mentioned that passwords should be strong, generated or unique per service. Despite generally good suggestions, a few suggested regularly changing passwords, which current research considers bad advice [59], [60]. Beyond these authentication measures, interviewees' suggestions became more diverse or vague. In about half cases, interviewees voiced a rather vague idea of security measures, likely coinciding with their sense of security: "*I feel like that's just the default or the baseline expectation. Same with, don't leave your screen unlocked, that kind of thing. It goes without saying maybe.*" (P10).

While most interviewees mentioned the use of encryption to secure their OSS devices, only a few recommended it. This might be due to encryption typically relying on system defaults or related to how it mainly defends against physical threats, that interviewees in general perceived as rarer and as something where resistance was futile (cf. Section 4.2).

> Summary: Challenges, Improvements, and Recommendations.Interviewees perceived their personal risk of an attack as overall low. Most commonly, interviewees named high amounts of typically unvetted dependencies as a challenge. Other challenges include how the security behavior of others is unknown or intransparent, different human factors such as issues due to abandoned projects or inexperienced developers, or technical obstacles. Common desired improvements included improving their authentication security, becoming more careful in general, or resolving previously mentioned technical obstacles. Finally, interviewees most commonly recommended the security measures they deployed themselves, such as using MFA or password managers.

5. Discussion

Below, we discuss our findings, make recommendations for individual security setups for OSCs, and contextualize our results with related work.

5.1. Deployed Individual Security Setups (RQ1)

Interviewees most commonly deployed measures regarding their authentication security, i. e., they used MFA and adopted password managers to benefit from long, complex, or randomly generated passwords. Furthermore, most used disk encryption to secure their devices. Beyond this, physical security was irrelevant for OSCs, as they did not expect physical threats. If they did, they expected attackers to be dedicated and violent, in which case physical security would have no effect.

In our interviewees' projects, there were rarely any secrets that required secure sharing. When necessary, interviewees typically used shared password manager vaults or external services or handled secrets within the CI. In this case, only specific authorized users had access to the secrets, using access hierarchies to decrease the impact of account compromise, as, e. g., unauthorized contributors could not trigger a package release.

Otherwise, we found that several measures were less common, including code signing, which was perceived as impractical and to some degree questionable as a security measure, ad blockers or trackers to protect against, e. g., malicious JavaScript, or Antivirus software. Overall, our interviewees were security-aware and tech-savvy. They typically put significant effort into their security setups and avoided deploying measures they deemed either less effective or lacking in usability.

5.2. Common Challenges among Contributors (RQ2)

We identified several challenges our interviewees perceived regarding individual security setups. First and foremost, they commonly mentioned issues surrounding the vast number of dependencies and, thereby unknown code within OSS, that were described as impossible to audit. While this was perceived as a well-known issue, interviewees criticized the lack of care, i. e., how nobody seemed to make an effort to decrease the number of dependencies, audit external software, or use dependency pinning to at least control which (safe) versions were imported by their projects.

Although our interviewees argued that conversations about security were rare, and in some cases mentioned that confronting others would make them feel uncomfortable, some also mentioned that this level of trust could lead to problems later on. We perceived that interviewees commonly projected their security-awareness onto others and assumed a high level of security as common sense and obvious. However, we only heard of OSCs questioning individual security setups when visual proofs of enabled MFA were available, which made the lack of it publicly noticeable. Overall, we argue this lack of discussion fuels the perception of security as a secondary task, and that more open dialogue and attention could mitigate this.

In several other cases, interviewees reported a lack of usability or technical issues, e. g., regarding MFA, code signing, or how local builds are handled.

Finally, while interviewees were security-aware, their perception of risk was low, either as they never really thought about potential incidents or because they did not perceive themselves or their projects as interesting targets for attacks [61], [62].

5.3. Better Supporting Contributors with Individual Security Setups (RQ3)

OSC tend to be pragmatic and only deploy a minimum of security measures that seem effective and easy to use. Below, we provide additional best practices or suggestions on how OSC could improve their individual security setups by discussing our observations and interviewees' suggestions.

Authentication Security: Especially seeing how many previous incidents happened based on weak authentication [14]–[17], the wide adoption of authentication security measures such as MFA and password managers can be considered an essential minimum. Our interviewees adopted both widely, suggesting that they are low-barrier security measures (cf. Section 4.2).

Platform-Enforced Measures: We observed a reluctance to discuss security among interviewees, which often included not wanting to pressure each other toward adopting specific security measures. However, our interviewees generally agreed with and accepted platform-enforced security measures, thereby removing the need to discuss or enforce the adoption of, e. g., MFA on a case-by-case basis (cf. Section 4.3). We, therefore, reaffirm current advancements by, e. g., GitHub [58], npm [63], or PyPI [64], that are gradually enforcing MFA for all accounts. In cases where these enforced measures are project- or organization-based, we recommend that project owners consider enabling them for all contributors.

Keep Enforced Requirements Inclusive: Related to the above security measures, we recommend that additional security measures should not decrease the inclusivity of OSS. Enforced measures should be low barriers and aim to not exclude interested and skilled contributors. For example, we recommend avoiding enforcing the use of hardware keys to not scare away contributors who might not be willing to deal with the usability issues [65]–[67] or associated costs [68], [69].

Manage Hierarchies and Access Rights: We recommend project owners to utilize clear hierarchies in their projects. While this can lead to a single point of failure when accounts are compromised, it also decreases the attack surfaces, as only specific individuals can cause severe harm with their elevated access and release rights. This was commonly highlighted as beneficial by our interviewees (cf. Section 4.3), and is utilized by larger projects such as the Linux kernel [70]. Using automated build processes, such as GitHub Actions, can allow OSCs to further manage access to code secrets without sharing them with other contributors [71].

Basic Device Security: Regarding devices, we suggest OSCs encrypt their devices and adopt a routine of locking them, not leaving them unattended, and only bringing them with them if necessary. We agree with the tendency that physical security is otherwise not as relevant as digital measures (cf. Section 4.2).

Virtualization to Separate Projects: We suggest utilizing virtual machines or containers to sandbox projects where possible [72], [73] as mentioned by some interviewees (cf. Section 4.5). This is especially relevant when contributing to multiple projects that could affect each other if one is compromised or when dealing with user-provided content, such as reproducible code snippets within bug reports.

Provide Guidance in Projects: Finally, despite the general high tech-affinity, we see a need for OSCs to discuss security (cf. Section 4.5). An excellent first step could be including some basic security measures such as the ones described above into existing guidance structures, such as contributing.md. These files are often included in OSS projects and contain details on how individuals can get involved with the project, and are an ideal place for suggesting measure for individual security setups of contributors. Since some interviewees were uncomfortable suggesting precise tooling choices, guidance could include basic measures such as MFA, password managers and password hygiene, device encryption basics, or even guides towards virtualization and containers when dealing with bug reports and external code snippets. Where feasible, we recommend providing a contact person who is willing to discuss security measures and answer questions to help kick-start more conversations regarding individual security setups.

5.4. Social Mechanisms Impact Individual Security Setups

In the following two sections, we discuss some of the most noticeable themes within our interview study.

A common tendency in our interviews was to report social mechanisms that influenced almost all contributor interactions regarding their individual security setups. For example, some OSCs reported to have quickly gained the trust of other project members and, therefore, were granted access rights quickly or were not removed from projects out of respect or not to antagonize others after stopping to contribute. This also impacted contributions, as about half interviewees mentioned that they did not need to write any security guidance because they trusted each other to have strong individual security setups without scrutiny. The same goes for the opposite side, as a similar amount of interviewees did not want to bring up a culture of distrust or paranoia, in which security was discussed or even examined: "*I don't want to come across as a paranoid person all the time. You'll talk about it less, even if maybe you're happy to be talking to a person who would want to hear about it.*" (P14).

This also carries over to programming practices, as one interviewee reported how the open source community knows that JavaScript has an issue with too many dependencies that are added too quickly to projects but that people did not speak up not to penalize or shame other contributors. Similarly, OSCs had a good sense of security but tried to avoid unnecessarily complicated measures, as security was only a lower priority. This is likely because the main goal of OSCs is the creation of software, and their security behavior and interactions are born from necessity, as P10 describes it:

> "*There is some desire, especially among developers, for the glory days of computing. [...] We could just write code and trust each other. I think still that's alive in the open-source [...].*"

5.5. Some Challenges Seem Hard to Solve

While we perceived a unique culture of trust, this also led to several noticeable drawbacks reported by our interviewees.

OSCs put a commendable lot of trust in each other and try to, e. g., avoid picking unnecessary fights by addressing potentially uncomfortable topics such as whether contributors use MFA. However, this can lead to a lack of guidance in practice, especially for junior contributors: "*Which is unfortunate, because it usually means that new people who contribute are just expected to somehow know or do things differently.*" (P14).

Although it seems to be hard to enforce security measures in OSS projects due to the freedom and lose hierarchies in projects, we argue that adding best practices to projects or providing more general security advice can be a good initial step to open up a dialogue about improved individual security setups.

In other areas, we found issues based on how quickly the OSS ecosystem welcomes contributors into projects, opening up the door for social engineering attacks. In this attack, malicious actors disguise themselves as well-meaning contributors, only to abuse access rights as soon as they are gained [74]. However, this is not an easy challenge to solve. While well-established projects such as Debian deploy an application process for new contributors, which requires new members to, e. g., show that they are experienced and have somebody vouching for them to be trustful and skilled [75], these processes are mainly unfeasible overhead for smaller projects. Additionally, requirements about previous experience can be an excluding obstacle for junior contributors who just started to become active in the OSS ecosystem. Similarly, there can be issues when project owners decide to stop doing OSS due to, e. g., a lack of time or funding, or from burning out, especially if they need to leave the projects behind without proper maintenance, or do not have the resources to correctly chose an inheritor. While there is no ideal solution for this problem, previous work stresses the importance of human factors, e. g., previous engagement or personal software need from other developers, for the survival of abandoned projects [38], [76], [77].

Overall, we find these issues more pressing in smaller projects, which do not have company backing or -funding, and rely on a few private contributors. In these cases, OSCs prioritize other issues over security and communication.

5.6. Putting our Findings Into Context

While Wermke *et al.* focused on a project perspective and mainly discussed other types of guidance and policies not focused on individual security setups of contributors, they found similar reasons as to why projects did not provide guides, including a lack of perceived relevance, time, or responsibility [23], which confirms previous findings on the lack of guidance or norms within OS projects [20]–[22]. Similarly, previous work has suggested using various forms of documentation or mentoring to mitigate these issues, especially concerning the on-boarding of new contributors [78]–[81].

Our extensive insights into the individual security setups and choices from OSCs extend previous work discussing challenges with large and intransparent dependencies [31], [38]. Fischer *et al.* discusses GitHub measures focused on code security and illustrates how platform-provided measures can help improve security within OSS [35].

Regarding the overall security mindset, previous work has shown that even expert users can tend towards less secure measures if they feel like the context justifies it, e. g., by using weaker passwords on lesser relevant websites, which fits the general practicality our interviewees displayed [33], [39], [44].

Finally, the importance of trust and social structures was also previously discussed [36], [82], [83]. Previous work suggests that high-quality commits, reputation of the employer, or personal connections are essential factors [84]–[86].

While previous work typically focused on code security and vulnerability disclosure, we provide insights into the multitude of individual security setups and choices made by OSCs. Our interviews give an extensive view and suggest that contributors are security-aware. However, they also uncover how more emphasis needs to be put on social interactions, especially the communication of (necessary) individual security setups within the OSS ecosystem.

6. Conclusion

In this work, we illustrated and discussed findings from 20 semi-structured interviews with OSCs from critical OSS projects. Most contributors were security-affine and deployed security measures they perceived as sensible, effective, and easy to use. However, beyond the usage of MFA and password managers, we perceived the deployment of other security measures such as device encryption or commit signing as less common. Individual security setups were only rarely discussed among contributors. Social mechanisms such as trust, respect, or politeness inhibited OSCs from enforcing or checking the adoption of security measures on other contributors. Overall, we make a list of recommendations to improve the status quo of individual security setups of contributors without impeding the unique culture of the open-source software ecosystem. We include suggestions for project owners, e. g., using platform-based security mechanisms instead of individually prompting contributors, utilizing access control hierarchies and CI/CD solutions to limit the number of accounts with privileged access or release rights, and discussing security more broadly, for example by adding individual security setup information for contributors to projects' contributing.md files.

Acknowledgments

We want to thank all of our participants for trusting us with their experiences and insights, thereby enabling us to do this research. We are grateful to the reviewers for their valuable feedback. We thank our lab students Stina Schäfer, Anne Vonderheide, Kateryna Nosik and Lukas Niehus for their early text drafts during our lecture. Additionally, we thank Nicolas Huaman, Niklas Busch, and Juliane Schmüser for aiding our work as backup interviewers and proofreaders. This research was funded by the VolkswagenStiftung Niedersächsisches Vorab – ZN3695 and by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy — EXC 2092 CASA – 390781972.

References

[1] Sonatype, *9th Annual State of the Software Supply Chain*, https://www.sonatype.com/press-releases/sonatype-9th-annual-state-of-the-software-supply-chain-report (visited on 12/03/2023).

[2] P. Cormier, *The State of Enterprise Open Source 2022*, https://www.redhat.com/en/resources/state-of-enterprise-open-source-report-2022 (visited on 08/02/2023).

[3] debug-js, *Network Dependents · debug-js/debug*, https://github.com/debug-js/debug/network/dependents (visited on 08/02/2023).

[4] NIST National Vulnerability Database, *CVE-2014-0160 Detail*, https://nvd.nist.gov/vuln/detail/cve-2014-0160 (visited on 08/02/2023).

[5] NIST National Vulnerability Database, *CVE-2021-44228 Detail*, https://nvd.nist.gov/vuln/detail/cve-2021-44228 (visited on 08/02/2023).

[6] NIST National Vulnerability Database, *CVE-2014-6271 Detail*, https://nvd.nist.gov/vuln/detail/cve-2014-6271 (visited on 08/02/2023).

[7] C. Falfe, *'Extremely bad' vulnerability found in widely used logging system*, https://www.theverge.com/2021/12/10/22828303/log4j-library-vulnerability-log4shell-zero-day-exploit (visited on 11/25/2023).

[8] S. Gallagher, *Heartbleed vulnerability may have been exploited months before patch*, https://arstechnica.com/information-technology/2014/04/heartbleed-vulnerability-may-have-been-exploited-months-before-patch/ (visited on 11/25/2023).

[9] npm Blog (Archive), *kik, left-pad, and npm*, https://blog.npmjs.org/post/141577284765/kik-left-pad-and-npm (visited on 08/02/2023).

[10] A. Sharma, *Dev corrupts NPM libs 'colors' and 'faker' breaking thousands of apps*, https://www.bleepingcomputer.com/news/security/dev-corrupts-npm-libs-colors-and-faker-breaking-thousands-of-apps/ (visited on 08/02/2023).

[11] A. Sharma, *BIG sabotage: Famous npm package deletes files to protest Ukraine war*, https://www.bleepingcomputer.com/news/security/big-sabotage-famous-npm-package-deletes-files-to-protest-ukraine-war/ (visited on 08/02/2023).

[12] T. Costa, *strong_password v0.0.7 rubygem hijacked*, https://withatwist.dev/strong-password-rubygem-hijacked.html (visited on 08/02/2023).

[13] I. Arghire, *Backdoor Found in 'rest-client' Ruby Gem*, https://www.securityweek.com/backdoor-found-rest-client-ruby-gem/ (visited on 08/02/2023).

[14] G. Thorpe, *NPM Library (ua-parser-js) Hijacked: What You Need to Know*, https://www.rapid7.com/blog/post/2021/10/25/npm-library-ua-parser-js-hijacked-what-you-need-to-know/ (visited on 08/02/2023).

[15] A. Sharma, *Popular 'coa' NPM library hijacked to steal user passwords*, https://www.bleepingcomputer.com/news/security/popular-coa-npm-library-hijacked-to-steal-user-passwords/ (visited on 08/02/2023).

[16] Henry, *Postmortem for Malicious Packages Published on July 12th, 2018*, https://eslint.org/blog/2018/

07/postmortem-for-malicious-package-publishes/ (visited on 08/02/2023).

[17] Gentoo Wiki, *Project:infrastructure/incident reports/2018-06-28 github*, https://wiki.gentoo.org/wiki/Project:Infrastructure/Incident_reports/2018-06-28_Github (visited on 08/02/2023).

[18] E. Holmes, *How I gained commit access to Homebrew in 30 minutes*, https://medium.com/@vesirin/how-i-gained-commit-access-to-homebrew-in-30-minutes-2ae314df03ab (visited on 08/02/2023).

[19] Python Security Documentation, *Account Takeover and Malicious Replacement of ctx Project*, https://python-security.readthedocs.io/pypi-vuln/index-2022-05-24-ctx-domain-takeover.html (visited on 08/02/2023).

[20] S.-F. Wen, "Learning Secure Programming in Open Source Software Communities: A Socio-technical View," in *Proceedings of the 6th International Conference on Information and Education Technology*, 2018, pp. 25–32.

[21] K. Constantino, S. Zhou, M. Souza, E. Figueiredo, and C. Kästner, "Understanding Collaborative Software Development: An Interview Study," in *Proceedings of the 15th International Conference on Global Software Engineering*, 2020, pp. 55–65.

[22] I. Steinmacher, T. Conte, M. A. Gerosa, and D. Redmiles, "Social Barriers Faced by Newcomers Placing Their First Contribution in Open Source Software Projects," in *Proceedings of the 18th ACM conference on Computer supported cooperative work & social computing*, 2015, pp. 1379–1392.

[23] 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 *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, IEEE, 2022, pp. 1880–1896.

[24] J. C. Santos, A. Peruma, M. Mirakhorli, M. Galstery, J. V. Vidal, and A. Sejfia, "Understanding Software Vulnerabilities Related to Architectural Security Tactics: An Empirical Investigation of Chromium, php and Thunderbird," in *2017 IEEE International Conference on Software Architecture (ICSA)*, IEEE, 2017, pp. 69–78.

[25] F. Zampetti, S. Scalabrino, R. Oliveto, G. Canfora, and M. Di Penta, "How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, IEEE, 2017, pp. 334–344.

[26] A. Gkortzis, D. Mitropoulos, and D. Spinellis, "VulinOSS: A Dataset of Security Vulnerabilities in Open-Source Systems," in *Proceedings of the 15th International conference on mining software repositories*, 2018, pp. 18–21.

[27] V. Piantadosi, S. Scalabrino, and R. Oliveto, "Fixing of security vulnerabilities in open source projects: A case study of apache http server and apache tomcat," in *2019 12th IEEE Conference on software testing, validation and verification (ICST)*, IEEE, 2019, pp. 68–78.

[28] R. Ramsauer, L. Bulwahn, D. Lohmann, and W. Mauerer, "The Sound of Silence: Mining Security Vulnerabilities from Secret Integration Channels in Open-Source Projects," in *Proceedings of the 2020 ACM SIGSAC Conference on Cloud Computing Security Workshop*, 2020, pp. 147–157.

[29] G. Antal, M. Keleti, and P. Hegedűs, "Exploring the Security Awareness of the Python and JavaScript Open Source Communities," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 16–20.

[30] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "SoK: Taxonomy of Attacks on Open-Source Software Supply Chains," in *2023 IEEE Symposium on Security and Privacy (SP)*, IEEE Computer Society, 2022, pp. 167–184.

[31] 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 *USENIX security symposium*, vol. 17, 2019.

[32] npm, *npm | Home*, https://www.npmjs.com/ (visited on 11/25/2023).

[33] N. Zahan, T. Zimmermann, P. Godefroid, B. Murphy, C. Maddila, and L. Williams, "What are Weak Links in the npm Supply Chain?" In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, 2022, pp. 331–340.

[34] 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: 17th International Conference, DIMVA 2020, Lisbon, Portugal, June 24–26, 2020, Proceedings 17*, Springer, 2020, pp. 23–43.

[35] F. Fischer, J. Höbenreich, and J. Grossklags, "The Effectiveness of Security Interventions on GitHub," pp. 2426–2440, 2023.

[36] M. Silic and A. Back, "Information Security and Open Source Dual Use Security Software: Trust Paradox," in *Open Source Software: Quality Verification: 9th IFIP WG 2.13 International Conference, OSS 2013, Koper-Capodistria, Slovenia, June 25-28, 2013. Proceedings 9*, Springer, 2013, pp. 194–206.

[37] K. Constantino, M. Souza, S. Zhou, E. Figueiredo, and C. Kästner, "Perceptions of open-source software developers on collaborations: An interview and survey study," *Journal of Software: Evolution and Process*, vol. 35, no. 5, e2393, 2023.

[38] D. Wermke, J. H. Klemmer, N. Wöhler, J. Schmüser, Y. A. Harshini Sri Ramulu, and S. Fahl., ""Always Contribute Back": A Qualitative Study on Security Challenges of the Open Source Supply Chain," in *Proceedings of the 44th IEEE Symposium on Security and Privacy (S&P '23)*, May 2023.

[39] E. Stobert and R. Biddle, "Expert Password Management," in *Technology and Practice of Passwords: 9th International Conference, PASSWORDS 2015, Cambridge, UK, December 7–9, 2015, Proceedings 9*, Springer, 2016, pp. 3–20.

[40] I. Ion, R. Reeder, and S. Consolvo, ""... no one can hack my mind": Comparing Expert and Non-Expert Security Practices," in *Symposium on Usable Privacy and Security (SOUPS)*, 2015.

[41] K. Busse, J. Schäfer, and M. Smith, "Replication: No One Can Hack My Mind Revisiting a Study on Expert and Non-expert Security Practices and Advice," in *Symposium on Usable Privacy and Security*, 2019, pp. 116–136.

[42] E. M. Redmiles, N. Warford, A. Jayanti, A. Koneru, S. Kross, M. Morales, R. Stevens, and M. L. Mazurek, "A Comprehensive Quality Evaluation of Security and Privacy Advice on the Web," in *29th USENIX Security Symposium*, USENIX, 2020, pp. 89–100.

[43] J. H. Klemmer, M. Gutfleisch, C. Stransky, Y. Acar, M. A. Sasse, and S. Fahl, ""Make Them Change it Every Week!": A Qualitative Exploration of Online Developer Advice on Usable and Secure Authentication," pp. 2740–2754, 2023.

[44] N. Lykousas and C. Patsakis, "Tales from the Git: Automating the Detection of Secrets on Code and Assessing Developers' Passwords Choices," in *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, IEEE, 2023, pp. 68–75.

[45] GitHub, *Security Overview · expressjs/express*, https://github.com/expressjs/express/security (visited on 12/03/2023).

[46] O. Dabic, E. Aghajani, and G. Bavota, "Sampling Projects in GitHub for MSR Studies," in *18th IEEE/ACM International Conference on Mining Software Repositories, MSR 2021*, IEEE, 2021, pp. 560–564.

[47] SEART, *GitHub Search*, https://seart-ghs.si.usi.ch/ (visited on 07/19/2023).

[48] GitHub, *GitHub Terms of Service*, https://docs.github.com/en/site-policy/github-terms/github-terms-of-service (visited on 11/25/2023).

[49] Qualtrics, *Qualtrics XM: The Leading Experience Management Software*, https://www.qualtrics.com/ (visited on 11/25/2023).

[50] Calendly, *Free Online Appointment Scheduling Software*, https://calendly.com/ (visited on 11/25/2023).

[51] V. Clarke and V. Braun, "Thematic Analysis," in *Encyclopedia of Critical Psychology*. New York, NY: Springer New York, 2014, pp. 1947–1952.

[52] ATLAS.ti, *ATLAS.ti | The #1 Software for Qualitative Data Analysis*, https://atlasti.com/ (visited on 11/25/2023).

[53] N. McDonald, S. Schoenebeck, and A. Forte, "Reliability and Inter-Rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–23, 2019.

[54] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. van Eeten, "A different Cup of TI? The Added Value of Commercial Threat Intelligence," in *29th USENIX Security Symposium (USENIX Security 20)*, USENIX Association, Aug. 2020, pp. 433–450.

[55] S. Höltervennhoff, P. Klostermeyer, N. Wöhler, S. Fahl, and Y. Acar, ""I wouldn't want my unsafe code to run my pacemaker": An Interview Study on the Use, Comprehension, and Perceived Risks of Unsafe Rust," in *In 32nd USENIX Security Symposium, USENIX Security '23, Anaheim, CA, USA, August 9-11, 2023*, USENIX Association, 2023.

[56] Department of Homeland Security (DHS)/Science and Technology Directorate (S&T)/Cyber Security Division, "The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research," 2012.

[57] FIDO Alliance, *Passkeys*, https://fidoalliance.org/passkeys/ (visited on 11/25/2023).

[58] GitHub, *Raising the bar for software security: GitHub 2FA begins March 13*, https://github.blog/2023-03-09-raising-the-bar-for-software-security-github-2fa-begins-march-13/ (visited on 11/25/2023).

[59] Y. Zhang, F. Monrose, and M. K. Reiter, "The Security of Modern Password Expiration: An Algorithmic Framework and Empirical Analysis," in *Proc. 17th ACM Conference on Computer and Communication Security (CCS'10)*, ACM, 2010.

[60] S. Chiasson and P. C. van Oorschot, "Quantifying the Security Advantage of Password Expiration Policies," *Designs, Codes and Cryptography*, vol. 77, no. 2, pp. 401–408, 2015.

[61] J. Mink, H. Kaur, J. Schmüser, S. Fahl, and Y. Acar, ""Security is not my field, I'm a stats guy": A Qualitative Root Cause Analysis of Barriers to Adversarial Machine Learning Defenses in Industry," in *In 32nd USENIX Security Symposium*, 2023.

[62] N. Huaman, B. von Skarczinski, D. Wermke, C. Stransky, Y. Acar, A. Dreißigacker, and S. Fahl, "A Large-Scale Interview Study on Information Security in and Attacks against Small and Medium-sized Enterprises," in *In 30th USENIX Security Symposium, USENIX Security '21, Vancouver, B.C., Canada, August 11-13, 2021*, USENIX Association, Aug. 2021.

[63] npm Docs, *Requiring two-factor authentication in your organization*, https://docs.npmjs.com/requiring-two-factor-authentication-in-your-organization (visited on 12/03/2023).

[64] D. Stufft, *Securing PyPI accounts via Two-Factor Authentication*, https://blog.pypi.org/posts/2023-05-25-securing-pypi-with-2fa/ (visited on 12/03/2023).

[65] J. Reynolds, T. Smith, K. Reese, L. Dickinson, S. Ruoti, and K. Seamons, "A Tale of Two Studies: The Best and Worst of YubiKey Usability," in *2018*

IEEE Symposium on Security and Privacy (SP), 2018, pp. 872–888.

[66] F. M. Farke, L. Lorenz, T. Schnitzler, P. Markert, and M. Dürmuth, ""You Still Use the Password After All" – Exploring FIDO2 Security Keys in a Small Company," in *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, USENIX Association, Aug. 2020, pp. 19–35.

[67] K. Reese, T. Smith, J. Dutson, J. Armknecht, J. Cameron, and K. Seamons, "A Usability Study of Five Two-Factor Authentication Methods," in *Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019)*, 2019.

[68] L. Würsching, F. Putz, S. Haesler, and M. Hollick, "FIDO2 the Rescue? Platform vs. Roaming Authentication on Smartphones," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, ser. CHI '23, <conf-loc>, <city>Hamburg</city>, <country>Germany</country>, </conf-loc>: Association for Computing Machinery, 2023.

[69] L. Lassak, E. Pan, B. Ur, and M. Golla, "Why Aren't We Using Passkeys? Obstacles Companies Face Deploying FIDO2 Passwordless Authentication,"

[70] Linux Kernel, *Linux Kernel Contribution Maturity Model*, https://www.kernel.org/doc/html/latest/process/contribution-maturity-model.html (visited on 12/03/2023).

[71] SLSA, *SLSA • Supply-chain Levels for Software Artifacts*, https://slsa.dev/ (visited on 12/03/2023).

[72] A. Khalimov, S. Benahmed, R. Hussain, S. A. Kazmi, A. Oracevic, F. Hussain, F. Ahmad, and C. A. Kerrache, "Container-Based Sandboxes for Malware Analysis: A Compromise Worth Considering," in *Proceedings of the 12th IEEE/ACM International Conference on Utility and Cloud Computing*, 2019, pp. 219–227.

[73] GitLab, *A beginner's guide to container security*, https://about.gitlab.com/topics/devsecops/beginners-guide-to-container-security/ (visited on 12/03/2023).

[74] npm Blog, *Details about the event-stream incident*, https://blog.npmjs.org/post/180565383195/details-about-the-event-stream-incident (visited on 08/02/2023).

[75] Debian, *New Members Corner*, https://www.debian.org/devel/join/nm-checklist (visited on 12/03/2023).

[76] G. Avelino, E. Constantinou, M. T. Valente, and A. Serebrenik, "On the Abandonment and Survival of Open Source Projects: An Empirical Investigation," in *2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, 2019, pp. 1–12.

[77] C. Miller, D. G. Widder, C. Kästner, and B. Vasilescu, "Why Do People Give Up Flossing? A Study of Contributor Disengagement in Open Source," in *Open Source Systems: 15th IFIP WG 2.13 International Conference, OSS 2019, Montreal, QC, Canada, May 26–27, 2019, Proceedings 15*, Springer, 2019, pp. 116–129.

[78] I. Steinmacher, C. Treude, and M. A. Gerosa, "Let Me In: Guidelines for the Successful Onboarding of Newcomers to Open Source Projects," *IEEE Software*, vol. 36, no. 4, pp. 41–49, 2018.

[79] S. Balali, U. Annamalai, H. S. Padala, B. Trinkenreich, M. A. Gerosa, I. Steinmacher, and A. Sarma, "Recommending Tasks to Newcomers in OSS Projects: How do Mentors Handle It?" In *Proceedings of the 16th International Symposium on Open Collaboration*, 2020, pp. 1–14.

[80] I. Steinmacher, T. U. Conte, C. Treude, and M. A. Gerosa, "Overcoming Open Source Project Entry Barriers With a Portal for Newcomers," in *Proceedings of the 38th International Conference on Software Engineering*, 2016, pp. 273–284.

[81] J. Dominic, J. Houser, I. Steinmacher, C. Ritter, and P. Rodeghero, "Conversational Bot for Newcomers Onboarding to Open Source Projects," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 46–50.

[82] W. Scacchi, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani, *Understanding Free/Open Source Software Development Processes*, 2006.

[83] K. Crowston, K. Wei, J. Howison, and A. Wiggins, "Free/Libre Open-Source Software Development: What We Know and What We Do Not Know," *ACM Computing Surveys (CSUR)*, vol. 44, no. 2, pp. 1–35, 2008.

[84] M. Antikainen, T. Aaltonen, and J. Väisänen, "The Role of Trust in OSS Communities - Case Linux Kernel Community," in *Open Source Development, Adoption and Innovation: IFIP Working Group 2.13 on Open Source Software, June 11–14, 2007, Limerick, Ireland 3*, Springer, 2007, pp. 223–228.

[85] V. S. Sinha, S. Mani, and S. Sinha, "Entering the Circle of Trust: Developer Initiation as Committers in Open-Source Projects," in *Proceedings of the 8th Working Conference on Mining Software Repositories*, 2011, pp. 133–142.

[86] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of Social and Technical Factors for Evaluating Contribution in GitHub," in *Proceedings of the 36th International Conference on Software Engineering*, 2014, pp. 356–366.