Developers' Approaches to Software Supply Chain Security: An Interview Study

Rami Sammak
rami.412222@yahoo.com
Paderborn University
Paderborn, Germany

Anna Lena Rotthaler
anna.lena.rotthaler@uni-paderborn.de
Paderborn University
Paderborn, Germany

Harshini Sri Ramulu
harshini.sri.ramulu@uni-paderborn.de
Paderborn University
Paderborn, Germany

Dominik Wermke
dwermke@ncsu.edu
North Carolina State University
Raleigh, NC, USA

Yasemin Acar
yasemin.acar@uni-paderborn.de
Paderborn University and George Washington University
Paderborn, Germany and USA

Abstract

Software Supply Chain Security (SSC) involves numerous stakeholders, processes and tools that work together to deliver a software product. A vulnerability in one element can cascade through the entire system and potentially affect thousands of dependents and millions of end users. Despite the SSC's importance and the increasing awareness around its security, existing research mainly focuses on either technical aspects, exploring various attack vectors and their mitigation, or it empirically studies developers' challenges, but mainly within the open source context. To better develop supportive tooling and education, we need to understand how developers consider and mitigate supply chain security challenges.

We conducted 18 semi-structured interviews with experienced developers actively working in industry to gather in-depth insights into their experiences, encountered challenges, and the effectiveness of various strategies to secure the SSC. We find that the developers are generally interested in securing the supply chain, but encounter many obstacles in implementing effective security measures, both specific to SSC security and for general security. Developers also mention a wide set of approaches and methods to secure their projects, but mostly report general secure software engineering methodologies and seem to be mostly unaware of SSC specific threats and mitigations.

CCS Concepts

• **Security and privacy → Usability in security and privacy**.

Keywords

Software Supply Chain, Software Security, Cybersecurity, Interviews, Developers

SCORED '24, October 14–18, 2024, Salt Lake City, UT, USA

ACM ISBN 979-8-4007-1240-1/24/10
https://doi.org/10.1145/3689944.3696160

ACM Reference Format:
Rami Sammak, Anna Lena Rotthaler, Harshini Sri Ramulu, Dominik Wermke, and Yasemin Acar. 2024. Developers' Approaches to Software Supply Chain Security: An Interview Study. In *Proceedings of the 2024 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED '24), October 14–18, 2024, Salt Lake City, UT, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3689944.3696160

1 Introduction

The Software Supply Chain (SSC) includes the processes, materials, and stakeholders involved in delivering software products or services to consumers. As part of this chain, using existing tools, resources, and infrastructure allows for cost and time effective software product development and distribution without having to create everything from scratch. Relying on external tools and libraries also means that the companies using them cannot directly control and ensure the security of these external components. In response to prominent attacks such as the SolarWinds Orion breach [56, 71], the NotPetya ransomware [49], and the recent xz-utils backdoor [18], a concerted effort by both governments [63, 64] and industry has been made to develop a set of guidelines and practices that address the growing threat of SSC attacks. Because of its complexity and many stakeholders, implementing SSC security is often not straightforward. Especially developers might face numerous obstacles and challenges, ranging from establishing trust for third-party software components to the difficulty of ensuring that external code components are vulnerability-free and up-to-date.

In this work we explore the current state of securing the SSC in industry, specifically by investigating how industry developers deal with the hurdles they face when trying to secure their SSC, analyzing the thinking process behind adopting certain security measures and neglecting others, and studying the factors that influence and the perceived effectiveness of the tools they employ.

RQ1: *How aware are developers of the Software Supply Chain?* We investigate if and how aware developers are of the SSC in general, processes, and defenses, including secure coding practices, consideration of human factors, sharing experiences, and informing themselves about current threats and technologies.

RQ2: *What challenges do developers face when trying to implement Software Supply Chain security measures?* SSC involves different

stages of security that vary depending on the type and complexity of the product and its environment. We investigate the challenges developers encounter when trying to secure their products and examine the reason behind them.

RQ3: *What tools or methods do developers use for Software Supply Chain security and what approaches do they consider to be effective?* SSC includes all stages from development to product delivery to the consumer, requiring security for each stage with appropriate methods. We aim to identify the security tools and methods developers use to secure their projects and to examine their experiences regarding the effectiveness of these tools and methods.

2 Related Work

We present related work in two areas: research literature in the context of SSC security and security studies involving interviews.

2.1 Software Supply Chain Security

The SSC consists of many different components and processes, resulting in the individual components being of interest to attackers and researchers alike including systematizations of knowledge [4, 52]. Ohm et al. analyzed 174 malicious software packages used in real-world attacks on open source SSCs, finding that 56% of packages triggered their malicious behavior on installation and 61% leveraged typo squatting [51]. Ladisa et al. presented a taxonomy of attacks on open source supply chains validated by user surveys with 17 domain experts and 134 developers [36]. Recent research also investigated SSC concepts like Software Bill of Materials (SBOM) [12, 50, 62, 77] and metrics like OpenSSF Scorecards [53, 75, 76, 78].

Dependencies act as 'links' in the SSC allowing projects to benefit from existing code, but can also be an entry vector for attacks and vulnerabilities. Research in this area include outdated dependencies [35], dependency selection [40, 73], typosquatting [48], and abandoned dependencies [44]. Yan et al. iteratively explored the attack surface of supply chain residual vulnerabilities in open source projects [74]. Updating and patching vulnerable dependencies is an important maintenance step. Pashchenko et al. investigated vulnerable dependencies in open source projects finding that the vast majority (81%) may be fixed by simply updating to a new version [54].

On the side of defenses for dependencies and updates, Ferreira et al. proposed a lightweight permission system to protect Node.js applications [15], Gonzalez et al. presented a tool to identify malicious commits [21], and Froh et al. proposed a differential static analysis approach to detect malicious code in package updates [19].

As part of the SSC, package repositories are a common data source for measurement studies, e.g., JavaScript's npm [1, 13, 42, 72, 79, 80], Python's PyPI [2, 67], Ruby's gem [30], and Google's Android [14]. Gu et al. conducted a one year measurement study spanning six registries and seventeen popular mirrors, covering over 4 million packages, finding that multiple threats exist in every ecosystem, and some have been exploited by attackers [24]. Ladisa et al. analyzed seven ecosystems to show how attackers use package managers and languages for arbitrary code execution in open source supply chain attacks, identifying 3 install-time and 4 runtime techniques [37].

Past research into the security of continuous integration and continuous delivery (CI/CD) and build systems include hardening [7], infrastructure as code [57, 58], and security strategies [11]. A number of scientific works focuses on GitHub action workflows [23, 33], including the creation of new analysis tools [9, 47]. A defensive build approach involves reproducibly building software artifacts or packages directly from source code repositories [22, 38, 68]. Protecting development environments is an important part of SSC security, including published research in the areas of code secret leakage (like API keys and passwords) [6, 34, 60] and IDE plugins [41].

Previous research in the SSC context mostly focuses on measurements and tooling; in this work we conducted interviews to investigate aspects that are not necessarily visible on a code level such as developers' awareness, perceptions, and encountered challenges in regards to securing the SSC of their products.

2.2 Security Interview Studies

Interview studies are a well-established, qualitative research approach for in-depth evaluations in the security research community, however, little research centers human factors along the entire SSC. Past interviews gained insights into the perceptions and work of experts such as security professionals [10, 59], administrators [8], app developers [65], and ML developers [45, 46]. Past studies also covered individual SSC technology topics such as open source components [69, 70], development processes and tooling [25, 29], programming languages [27], cryptography and authentication [16, 26, 32], reproducible builds [17], and dependency selection [39, 55].

In 2024, Amft et al. conducted 20 semi-structured interviews with experienced open source software contributors, finding that despite a high affinity for security, contributors face challenges due to heterogeneous security setups, lack of enforced guidelines, and social factors like trust and respect that hinder the sharing of security knowledge and best practices [3]. In a 2024 preprint, Kalu et al. conducted interviews with 18 high-ranking industry practitioners across 13 organizations, finding that while software signing is recommended for improving supply chain security, its adoption is hindered by technical, organizational, and human challenges [31].

As shown by these related works, interviews are a well-established method in security and privacy research, allowing researchers to effectively gather in-depth insights from software experts. While recent research focuses on individual components of the SSC, such as integrating [69, 70] and signing components [31], or building reproducibly [17], our research investigates how developers holistically consider security in the SSC. In a holistic overview we explore how developers consider supply chain security practices pre-development to post-deployment, by interviewing them about all aspects and measures during all development stages, rather than individual processes as in prior work.

3 Methodology

In this section, we describe our research approach and the design of the semi-structured interviews. To explore SSC security issues, the challenges developers face, and effectiveness of various security measures employed within commercial industry, we conducted 18

semi-structured interviews with software developers and engineers actively working in the field between December 2023 and February 2024.

3.1 Interview Guide

We developed an initial interview guide based on our research questions and extended it based on relevant related work. Some questions requiring clarification were accompanied by explanatory notes designed in such a way that they would not influence the course of the interview or create bias. We conducted two pilot interviews with software developers. Based on feedback, we rearranged questions between sections and rephrased others. The final interview guide is provided in Appendix A.

3.2 Recruitment

Our recruitment strategy focused on currently employed software developers from various industries with over three years of commercial experience to ensure participants had experience with SSC concepts or at least some exposure to them. We utilized multiple recruitment channels (personal networks, discord, telegram) to maintain a balance between quality and accessibility of participants. Specifically, we looked for developers who satisfy our inclusion criteria through our professional network, Discord channels, and IT Telegram groups, where we posted messages that contained details about our study, supplementary information, and contact details. We aimed to recruit developers from a diverse range of backgrounds, given that security measures may vary by country and industry, e.g., financial institutions and telecommunications may require specific security protocols and measures by local laws. We chose recruitment channels to reach our desired sample, and did not access any pre-existing pool of participants. Most of those in our professional network we recruited opted in. As we also recruited through open channels, we cannot make statements about opt-in rates, as we do not know how many potential participants saw our messages, and chose not to participate. We did not pay participants for participation.

3.3 Interview Procedure

All the interviews were conducted through our university's Zoom instance between December 2023 and February 2024. Before starting the interview, the participants were briefed about its purpose and goals. It was clarified that participation was voluntary, that respondents were free to ask for clarification should a question be confusing, and that they were free to skip any question. Interviewees were promised anonymity and were asked for consent to audio record the interviews, emphasizing that the recordings will be destroyed after transcription. Interviews lasted between 35 and 50 minutes. The semi-structured interviews were guided by the major topics illustrated in Figure 1.

3.4 Data Analysis

A codebook was created to systematically define and catalog the various themes and concepts that emerged during the coding process. Each entry in the codebook consisted of a code, its description, and potentially a list of subcodes that were derived from the resulting data.

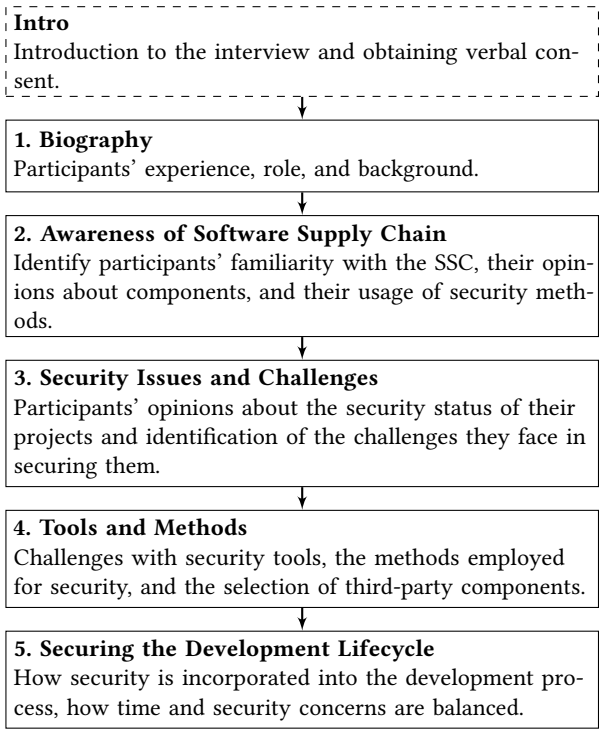

Figure 1: Illustration of the flow of topics in the semi-structured interviews. In each section, participants were presented with general questions and corresponding follow-ups, but were generally free to diverge from this flow at will.

During the analysis, both thematic analysis and content analysis were employed. The former was used to conduct an in-depth analysis of the responses to open-ended questions where respondents shared their personal opinions. This method helped identify key patterns in participants' views, while the later one helped analyze more structured responses, such as determining the frequency of mentions of different tools or methodologies. This approach provided a comprehensive understanding of the studied issues, combining both qualitative and quantitative aspects of the data.

3.5 Ethical Consideration

In this study, we adhere to the best practices from the Menlo report such as beneficence, respect for persons, respect for laws, and justice. Prior to signing up, we provided the participants with detailed information about the study's aims, its procedures, and the handling of the collected data. This approach ensured that participants were well-informed and could make a knowledgeable decision about their involvement. We encouraged potential participants to ask questions and informed them that their participation was entirely voluntary. Prior to the interviews, we obtained participants' informed consent and made them aware of their right to skip any question for any reason, whether due to lack of knowledge, preference not to disclose, or restrictions on revealing certain information. We also informed them of their freedom to withdraw from the study at any moment without any consequences.

All collected data was managed, processed, and stored in strict compliance with GDPR requirements. Upon completion of transcription, all original interview recordings were destroyed to further ensure participant confidentiality. Participants were provided with contact information should they have any follow-up questions or require further clarification after the interviews. All authors involved in data collection and analysis were enrolled/employed by an institution outside the US that does not require ethics review for this type of research. While we did follow all best practices, we did not undergo formal review. We took great care to follow research best practices, as well as principles outlined in the Menlo Report [5].

3.6 Limitations

This study includes several limitations that should be considered when interpreting its results. The sample size of 18 developers may not fully capture the diversity of SSC practices and challenges on the large scale of the software ecosystem. The reliance on qualitative, self-reported data introduces a degree of subjectivity and potential bias, as participants' responses are influenced by their individual experiences, knowledge, and perspectives. Furthermore, the rapidly evolving nature of technology, security practices, and emerging SSC frameworks means that some findings might quickly become outdated. The focus solely on developers may also neglect other key stakeholders in the SSC, such as security specialists, project managers, QA engineers, and senior executives, whose approaches to security could offer additional valuable perspectives. Lastly, confidentiality concerns may have restricted the level of detail participants shared about proprietary technologies or sensitive security measures, potentially limiting a thorough grasp of SSC security operations in industry. To protect participants' identities and workplace privacy, we agreed with them not to publicly share full transcripts.

4 Results

We present the results of 18 semi-structured interviews with participants involved in software development. We explored their awareness and perceptions regarding SSC security, and we found that developers do not consider SSC security holistically. However, participants mentioned considering various aspects of general software security and open source software security.

4.1 Participants

We interviewed 18 participants, 14 of whom hold degrees in computer science or computer engineering. Their industry experience ranges from three to over ten years, with an average of eight years. The participants' professional roles include freelancing, web development, DevOps engineering, embedded systems, project management, and cloud technology. They have worked in diverse industries such as financial technology, healthcare, sports, e-commerce, automotive, and government. An overview of participants' demographics is shown in Table 1.

P.No	Position	Industry	Country
P01	Senior DevOps	Telecommunication	Germany
P02	Lead Developer	IT Services	India
P03	Lead Developer	IT Services	India
P04	Senior Developer	IT Services	Chile
P05	Developer	IT Services	Germany
P06	Senior Developer	Healthcare	Germany
P07	Developer	Telecommunication	Sweden
P08	Lead Developer	Manufacturing	India
P09	Senior Developer	Various	UAE
P10	Senior Developer	IT Services	Germany
P11	Software Architect	Banking	USA
P12	Senior Developer	Healthcare	Pakistan
P13	Software Architect	IT Services	India
P14	Senior Developer	Various	India
P15	Senior Developer	IT Services	Pakistan
P16	Lead Developer	IT Services	Germany
P17	Developer	Sports	Russia
P18	Developer	Banking	Germany

Table 1: Participants' demographics based on self-reporting in the interviews. Position and Industry are binned into categories to protect participants' identities.

4.2 Developers' Awareness of the SSC

Out of 18 participants, 11 expressed some familiarity with the concept of the SSC. However, despite this general awareness, some participants encountered challenges in formally speaking about SSC, and were unsure on what it formally describes and encompasses. While most of them understood its various elements and were relatively accurate in discussing its essence, they expressed doubts about the reliability of their knowledge, e.g., "*I'm somewhat familiar with software supply chain that is the end to end delivery from software initiation till the software delivery, am I correct?*" (P02). Participant P16 also directly pointed out the problem of lack of formalization of the concept: "*We do not formalize these processes so much.*" (P16) Participant P17 expressed difficulty in providing a concrete definition, but highlighted its application in their professional activities: "*Maybe I don't know the name, the actual name, but I do know it in practical way, I think.*" (P17)

Participants are aware of the elements of the SSC but not the term. For participants who initially struggled to fluently discuss the SSC, we shared an explanation in Zoom's interview chat box, based on prior literature [20, 28, 43], as follows: "*Software supply chain refers to the entire process involved in creating and delivering software products. Main components found in almost every software supply chain:* ***Source code repository*** *(GIT, SVN etc.),* ***Dependencies and libraries****: third-party libraries and open source dependencies,* ***CI/CD systems****: Jenkins, Gitlab CI/CD, azure devops etc.,* ***Development tools****: IDEs, code editors, Version Control, building tools (e.g.: apache maven, webpack etc.),* ***Deployment****: Docker, cloud, databases, Testing: unit testing, security testing (pentesting, authentication and authorization testing, etc.),* ***People****: developers, QA testers, project managers, open source maintainers, product owners,* ***Code scanning and analysis***". After reading our explanation, some participants noted that they were familiar with many elements of the chain, but had trouble articulating the concept: "*So I do all those things, but*

I'm not aware of the software supply chain. Yeah. But I do all those things and do not know the word." (P10)

Participants' understanding of SSC security is shaped by a combination of formal education and practical experience. However, despite the general awareness, there is a gap in the accuracy and formalization of their understanding of the concept. Participants possess practical experience in interacting with the different SSC components but face difficulties in fluently articulating their place in the SSC. Overall, even though most participants have at least heard of the term, some indicated that they have a limited understanding of the SSC and do not holistically consider it in their development processes.

Low awareness of supply chain attacks. Participants did not report having been directly targeted by supply chain attacks. When asked, they either volunteered issues not attacking the supply chain, or reported patching well-publicized supply chain issues along with other industry players, as detailed by participant P10:

> "*Normally these kind of issues comes with when we discover new kind of vulnerabilities in our third-part library or some kind of tool that we are using. Common example I can give you like this or like local Log4J issue that was in a Java library. That kind of issue sometimes comes in. Yeah, that's like common thing happened to us. […] we have addressed that critically and immediately.*" — P10

4.3 Security Issues and Challenges

We present security issues and challenges related to tools and components, and organizational challenges with budget, time, and resources. Our interviews highlight that participants report facing multiple challenges regarding general software security that often influence supply chain security.

4.3.1 Challenges with Tools and Components. Participants reported challenges with security tools and external components; they highlighted that issues like poor usability make it difficult to detect vulnerabilities. They further mentioned challenges with updating vulnerability scanning tools, making it challenging for them to detect new threats or worse—leading to them abandoning the tools.

Usability issues with security tools. Multiple participants reported challenges with the usability of security tools; specifically, one participant (P08) expressed frustration with a software composition analysis (SCA) security tool not predicting and reporting all issues in early runs, leading to a fractional approach to addressing vulnerabilities and issues being reported after initial fixes:

> "*So the main challenges that I see with the security tools is they cannot predict, at least they cannot report all the issues in the early runs of our security violation. On every run they keep reporting different issues. So for example we use [security tool] and when I run it now, if I get a report that it has some issues, the developers fix them, but when we run it after two weeks or three weeks later, it will report some issues which it should have reported earlier, because the code base did not change […]*" — P08

Participant P01 shared challenges associated with configuring certain security tools within the CI/CD pipeline when managing projects with different programming languages. They report that the primary issue lies in the complexity of configuring a static application security testing (SAST) tool they use so that it accurately scans projects based on the programming language they use: "*[…] imagine having a repo with many languages, each time you scan, you must edit the configuration file. This is one challenge.*" (P01) Such an issue is an example of the practical aspects developers face when integrating security measures into a project that uses a diverse set of technologies.

Participant P18 mentioned a problem they encountered after integrating a SAST tool into the CI/CD pipeline where it significantly increased the project build time: "*I remember once we had a problem where we tried to bake in the tool into our pipeline, but the build time became very long. I guess because of the codebase. It actually became so long that it affected our sprints.*" (P18)

Using third-party components introduces security risks. Some participants (8) mentioned challenges in balancing the usage of third-party components and the need to ensure security. Participant P17 talked about their decision to stop using certain frameworks due to their notable security flaws in an effort to maintain the overall project security: "*I can say I stopped using some frameworks because of security problems. Like [content management system], for example. [Content management system] is so easy to hack, I can say. So much information by default is given through API.*" (P17) Participant P01 discussed a critical challenge that could be posed by third-party libraries that are no longer updated, leaving known vulnerabilities unaddressed, and mentioned having "*[…] to live with this latest version that is vulnerable*" (P01)

Participant P10 mentioned issues of third-party components ceasing to provide support after a certain period of time, forcing the search for alternatives that meet the project standards. They mention the difficulty in finding suitable replacements, often leading to the development of in-house solutions that may greatly impact the speed of delivery:

> "*Sometimes projects or libraries don't provide support to a certain level after a certain time period. So in that case, if we don't have the support, we have to find an alternative for that. So there are alternatives, but sometimes it can be quite hard to find alternative that matches our requirements and our security standards, and that's where you have to use an engineering of your own, and that takes some additional time.*" — P10.

Participant P11 highlighted a major challenge they faced when a long-used open source software component underwent ownership changes, with another company becoming the new owner. This led to a major overhaul and the release of a new version, disrupting the project roadmap and highlighting the potential challenges that come with relying on third-party open source software. They mentioned that this challenge is " *[…] like that sort of throw a trainwreck into your roadmap.*" (P11)

The period during which a project is stuck on an outdated version of a component may expose it to security vulnerabilities, especially when projects are using old dependencies' versions that are no longer patched. Furthermore, integrating new components often

involves a reevaluation of security protocols followed within the project. All of this requires time, which may be utilized by malicious actors until a project is finally updated, as outlined by one of our participants: "*Sometimes we have to make those changes because for example once you're switching to something new, the steps we follow or the procedures we follow need to be adjusted accordingly, otherwise everything will get more complicated.*" (P10)

Although one of the most important security measures is auditing dependencies and keeping them up-to-date, the majority of participants (13) mentioned it as one of the most challenging security measures to implement. One participant mentions that this comes from a lack of financial resources: "*I believe nobody except the banks and the military and some other high level organization does a third party auditing to every component, and updates everything. I think that's way expensive. So it's very difficult for people to do that.*" (P04) Another reason could be compatibility issues that may be introduced upon updating: "*Also updating is challenging because when you update to the latest version you may need to change some parts of the code which could lead to code breaking.*" (P08) Additionally, participant P08 also adds that business needs can prevent updates: "*And if our business is getting solved even if the library has an old version, we will go with it to avoid downtime.*" (P08)

4.3.2 Organizational Challenges with Budget, Time, and Resources. Participants highlighted challenges with the lack of budget allocated for security, which makes it challenging to prioritize security. Additionally, they also explained that there is a lack of focus on security training; if they exist, they are typically generic and mostly also do not focus on SSC.

Lack of budget for security. Participants expressed major challenges with security tools and components, including configuration difficulties, integration issues, and the need to stay alert to prevent vulnerabilities. Furthermore, the task of managing, auditing, and updating third-party dependencies presents another significant challenge in software security. It is additionally hindered by various constraints, from financial limitations and compatibility issues to completing business priorities. The fear of introducing compatibility problems or disrupting business operations further complicates this challenge. Participant P11 expressed frustration that they believe that organizations mainly focus on profit, with security oftentimes not being prioritized:

> "*I think updating dependencies and validating and signing builds are probably really hard to implement. […] It's really because most companies already have a really long backlog of things to do for profit-generating, you know, market-based sort of thing, and their first target is not security*" — P11

Additionally, only 9 participants receive financial support from their employers to participate in courses, seminars, and workshops. This indicates that many participants do not receive sufficient support for professional training, especially in such a critical area such as software security.

Insufficient security training: Learnings are motivated by personal interest. Participants expressed a lack of support from organizations to further their knowledge in security. Further, participants expressed a commitment to self-education and efforts to keep their skills up to date despite the lack of direct motivation or financial support from employers. Many participants (10) self-initiate their own learning process by utilizing various online courses, forums, podcasts, and other resources to expand their knowledge independently. For instance, participant P15 gained hands-on experience in penetration testing and specified educating themselves about recent vulnerabilities.

Participant P17 mentioned that even though security-related tasks are part of their daily work, their superiors do not actively encourage further qualifications in the security domain, leaving them to pursue security skills independently:

> "*Yes, I do learn to get better knowledge on security, but for personal interests, not at the company level […] we always use security tools to protect ourselves and create our projects using some famous frameworks that also include in the background all these securities. But there are no trainings on how to use them in the company.*" — P17.

Many participants prefer to invest in training that directly relates to their current projects and professional goals. This reflects a pragmatic approach to career development that prioritizes skills that directly impact the workplace: "*You know, I would say since my job is more focused on DevOps and cloud computing and that stuff, I would say I would focus more on taking courses that are beneficial to this domain. […] Maybe in the future I might consider taking security courses.*" (P01)

Companies' security trainings are general and abstract. Some participants (3) whose companies offer internal security courses mentioned some issues within those courses. The issues centered around security training being too general and not practical to apply in their actual projects. For instance, participant P16's company provided a security course; the knowledge provided was perceived as abstract, and the program lacked the flexibility to adapt to the specific needs of employees:

> "*[…] they [security trainings] can be too abstract in general. And specifically the disadvantage that I saw in it was that the things that were discussed there, I couldn't apply to my project. And it would be much more useful if a person would come to our project and see how our processes are built and offer something of their own.*" — P16

Lack of dedicated security roles. Only five participants mentioned having security roles in their organizations that are either filled by a security professional or an entire unit responsible for managing security issues: "*We do have specific roles called security officers. So they can be security officers of different experience level. Some of them are at a software developer level. Some of them are at a architect level. Some of them are a principle architect level.*" (P08)

The allocation of security roles depends on many factors, including the company's budget, project complexity, and the level of leadership responsibility. In certain cases, these roles may be primarily focused on conducting penetration tests, which cover only some aspects of SSC security: "*Yes, for example, in our recent project, there was a team of the pen testers that would frequently check the security of the product and we would get the reports. […] but I don't think that would be enough for perfect security.*" (P15)

Other companies adopt a more comprehensive approach that includes training for developers: "*We have security in our team, security team and security master, I think it's called. We have that in every team. So one person, he got additional training for security within the company to analyze if there is any security risks that we introduce through our code changes.*" (P07)

Only four participants indicated having security roles and high security requirements within their companies, and felt that security was systematically pursued.

Navigating trade-off between security and project deadlines. Another major issue many participants frequently face is finding a balance between time pressure and the need for security. Participants report that often developers must make difficult decisions on whether to prioritize project timelines or address security concerns at the cost of delaying its release. Most participants (11) reported that due to strict time constraints within their company, they had to either postpone addressing certain security issues until after product release, or ignore them entirely:

"*We also try to publish the project so quickly because we always have a short deadline. That's why sometimes we publish the projects and push the security updates after the deadline.*" (P17)

Further, participant P03 expressed that their clients "*[d]on't want to make different adjustments with their deadline*" (P03) and mentioned that implementing comprehensive security measures can sometimes slow down the development process, leading to choosing less secure options to deliver on time: "*When you are implementing any big security scheme into the system, it's very difficult to meet the deadline so we have to cross those bridges and start using less secure things […]. People are just looking for the solution not looking for security sometimes.*" (P03)

Though a majority of participants grappled with deadlines and security, a few participants (5) mentioned taking a more systematic approach when faced with the issue of weighing time against security. Specifically, participant P02 mentioned that they make decisions based on the severity of the security issue, and try to postpone deadlines where necessary. "*When we are talking about security, there are two levels to it. One is critical and risky and then we have low security issues. So based on the priority of the security issue, we make the decision of whether to patch it or whether to release it.*" (P02)

Almost all participants mentioned cases where they had to balance tight deadlines with the need for strong security. They expressed that this issue mainly arises from the pressure of managers and clients who often care for features rather than security.

Balancing (mis)trust and security. Many participants (10) mentioned people (developers, users, and adversaries) as the most vulnerable component of the SSC. This includes a range of security factors, ranging from accidental errors to malicious actors. Trust was one of the most frequently mentioned reasons for considering people as the most vulnerable component: "*The most vulnerable component is people. Because relying on trustworthy developers is very hard these days. It's very hard to find a trustworthy developer with ethical and moral skills.*" (P09)

Participants expressed a general mistrust in developers, especially based on seniority level. Participants P16 and P13 reported feeling uncomfortable with junior developers having privileges and access to sensitive parts of the project. Participant P16 exclaimed their mistrust in junior developers by stating: "*Well, in my first company security was built on trust. That is, you need something, you just get maximum access to it. Even juniors. At some point you have access to delete all the resources. And it doesn't really scare anybody, it's just like people are betting that it's never going to happen.*" (P16)

Further, participant P17 pointed out that they fear potential sabotage or leaks of internal policies when certain team members leave the company under unfavorable terms. This highlights the need to strictly monitor not only the technical aspects of the project, but also the human aspect: "*[…] if someone leaves the company, maybe they didn't get paid, they might create so many problems for the team, like leaking company code or security rules.*" (P17)

Overall, many participants consider human factors as a critical vulnerability in software security, with challenges ranging from managing developer privileges to ensuring proper security training.

Security guidelines are not project specific. Some participants (8) reported having standards or guidelines related to security. However, despite their existence, they do not always meet the needs of teams and developers. Some participants pointed out that these guidelines are often not too useful: "*There are guidelines, yes. I don't think it's as detailed as coming down to specific rules. Sure, there are some rules in place, but most of the guidelines are very general and not really applicable to everyone.*" (P07)

Others find that the standards and practices in place can be quite beneficial. The effectiveness and applicability of these guidelines can vary significantly from one company to another, with some companies managing to create security guidelines that participants find useful: "*Yes. We have best practices and coding standards. So those documents are shared with all the developers. So whenever we feel that we are supposed to raise a full request for a certain feature, then we check that document.*" (P02)

Participant P17 stated that they "*[…] do not have these kind of security standards or policies*" and took the initiative in introducing guidelines within their company, leading their company to "*taking security seriously*". Participants' experiences with security standards and guidelines greatly varied in our interviews, with some finding them too general and not particularly useful, while others benefit from well-defined practices and coding standards. Some also took the personal initiative to introduce their organizations to guidelines.

4.4 Security Tools and Methods

In this section, we present participants' views on security tools and vulnerability assessment strategies. When asked about the tools and methods they use for SSC, participants expressed strategies they use for general software security, further highlighting a lack of awareness about SSC.

Use of security tools despite challenges and issues. Almost all participants specified employing different tools and methods to ensure project security. Participants mentioned using static code analysis, manual testing, keeping all third-party libraries and dependencies up to date, and software updates to eliminate vulnerabilities. Furthermore, some participants (7) emphasized the importance of continuously monitoring deployed systems to detect anomalies and promptly responding to security incidents: "*Continuous monitoring of deployed software. Deployed software means it is right now*

in the production stage mostly. And this is the code base which is being served to all of the end customers. So continuous monitoring is extremely important. […] Deployment phase is one of the most important phases." (P13)

Dependency auditing is another method mentioned to ensure security, despite the difficulties of conducting continuous dependency audits in projects due to time and resource constraints, it is an integral part of securing the SSC. The primary auditing tools that were mentioned include SonarQube (9), MEND, Black Duck, and OWASP dependency check, all of which aid in identifying and mitigating potential security threats in project dependencies. Even in situations where automated auditing tools are absent, participants usually resort to manual security testing: "*I would say that because also we don't have some kind of automated testing tools, so the reviews are mostly done manually.*" (P05)

Only some teams conduct security tests at different stages of the development process. Some participants (6) also reported conducting penetration testing, where penetration testers simulate attacks on the application and provide a report with patch requirements: "*The internal security team try to hack into the application using different tools and get a report which they send us, and we have to fix certain things.*" (P02)

During the software development lifecycle, participants specified the development and testing stages as the ones where they focus the most on security: "*The most fundamental phase where you should focus on security is development phase. And before you do the deployment, you apply a set of penetration testing, like trials and quality assurance to maintain and check if your code is 100% reliable for production.*" (P09)

Key factors for selecting security tools: Popularity, latest updates, user engagement, and known issues. All participants reported assessing a project's popularity and its user base, making this one of the most frequently mentioned verification methods. The popularity of a project can indicate that it has regular security checks and updates, which may imply that problems are detected fairly quickly: "*Well, in any case, you would have to check the libraries you're using, but I would say that if you're using a very famous, well-known library, it's more likely that it will be free from vulnerabilities because it's being used by many developers in many other projects.*" (P01)

Further, 15 participants mentioned verifying when a particular tool or component was last updated, making this the second most mentioned vulnerability check method:

> "*If any tool was updated more than a year ago then we never use it, because if that tool has not been updated in a long time, then there is a chance that it won't be updated in the future also and if there is any vulnerability or any security issue that may come in future there is no guarantee that it will be updated*" — P13

Additionally, few participants (4) go even further and check a project's known issues: "*If you are using GitHub, for instance, you can track all the issues in the issue section. From there, you can see what the community members are reporting and the vulnerabilities which are discovered on the repository itself.*" (P09) Participant P11 mentioned that their security teams perform an analysis of a project before allowing it to be integrated: "*We submit a request that we'd like to include this library. And in in a matter of days and weeks that [the security team] will come back and say yes or no.*" (P11)

Overall, participants employ a variety of tools and methods for security and vulnerability assessment, including static code analysis, manual testing, continuous monitoring, dependency auditing, and software updates. Participants also mentioned considering key metrics like popularity, engagement, known issues, or update frequency when selecting security tools.

No adoption of SBOMs. Participants were recruited internationally at a time that SBOMs were not yet widespread. Consequently, participants either did not know the term, or reported a more general strategy for keeping track of components and dependencies, if at all. For example, SBOMs will be required to comply with the upcoming EU cyber resilience act—this is, however, not in effect at the time of writing this paper, and apparently has not made it into interviewees' development practice yet.

4.5 Developers' Security Recommendations

In this section, we highlight participants' recommendations and suggestions for better security. Even though asked specifically about supply chain security, results highlight an overall need for software security, with less focus on SSC.

Developers should be aware of vulnerabilities. Participant P13 emphasized the importance for developers to have a comprehensive knowledge of the different vulnerabilities, especially with regard to the tools that they use: "*The developers should be very well aware of all of those security vulnerabilities and all of those development practices, that should be followed for technologies they use.*" (P13)

Developers should follow secure coding practices. Participants P02, P07, P14, and P18 point out the necessity of following secure coding practices: "*Secure coding practices are very important, actually, if you follow them, you will secure yourself against many OWASP listed attacks.*" (P14)

Developers should carefully select third-party components. Besides, they state that it is important to carefully select third-party components. Participants P01 and P11 recommend utilizing well-known and reliable repositories and software, stressing the importance of verifying the security credentials of third-party vendors: "*Use really well-known repositories or software. Don't just import or use any libraries or tool that is lesser known, including in your IDE environment and your development tools.*" (P11)

Developers should carefully manage access rights. Lastly, participants P12 and P16 discuss the importance of development environments and managing access rights. Participant P12 recommends creating distinct development environments to protect against unauthorized access, while participant P16 talks about protecting project resources from misuse, especially by new developers: "*One of the most important is probably the protection of resources. That is, which are involved in the project. This starts with access to the source files. So that a junior could not conditionally come in and destroy the database on the first day.*" (P16)

Our participants' recommendations highlight the complexity of managing the security of the SSC, including recommendations ranging from developers' education, careful tools selection, trusting people, and ending with following basic security practices.

5 Discussion

In this work we reported insights from 18 semi-structured interviews with experienced industry developers to formulate key recommendations for improving SSC security.

We find that developers care about SSC security, although not all of our participants were initially familiar with the specific term. We also find both a lack of awareness of specific SSC attacks and also a lack of knowledge around mitigations for these attacks. While our participants were aware, mentioned, and implement security practices and were securing open source components—an integral part of SSC security—we encountered no direct discussion of some other important SSC security approaches like secure builds and the prevention of certain deployment and runtime threats. Similarly, interviewees were not personally familiar with SBOM and supply chain attacks, and the Supply-chain Levels for Software Artifacts (SLSA) framework was also never discussed.

We think this shows that more awareness work, education, and engagement is needed about SSC security in industry, specifically highlighting the unique threats and mitigations in the context of the SSC. We also think that threat modeling with developers, as well as educational outreach in industry, can help increase awareness and skills for developers to protect the SSC. Finally, as we once again observe the importance of open source security for supply chain, we recommend efforts that secure and make transparent the security of open source components. Our work identified several issues like usability issues with existing security tools, we implore future work to determine whether these stem from a lack of training, insufficient documentation, or other user experience issues.

Based on our work, we recommend that the research community as well as industry leaders that demonstrably have expertise in securing the supply chain [61, 66] engage with a broad audience of developers to disseminate current advances in supply chain security.

6 Conclusion

We conducted 18 in-depth, semi-structured interviews with experienced software developers from industry between December 2023 and February 2024 to explore their experiences and challenges in securing the SSC. Our investigation focused on their awareness of SSC, the hurdles they encounter, and the effectiveness of the strategies and tools they employ. We discovered that while developers are generally aware of SSC's importance, they face significant obstacles in implementing effective security measures. These include issues with security tools, dependency management, and balancing security with development time. We also find a lack of awareness of new attacks and also defenses specific to supply chain security.

Acknowledgments

This work is supported in part by NSF grant CNS-2207008 and CNS-2206865. Any findings and opinions expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. We want to thank all interviewees for their participation and appreciate the knowledge and valuable time that they have generously given. We also thank the anonymous reviewers for their valuable feedback.

References

[1] Rabe Abdalkareem, Olivier Nourry, Sultan Wehaibi, Suhaib Mujahid, and Emad Shihab. 2017. Why do developers use trivial packages? an empirical case study on npm. In *Proceedings of the 2017 11th joint meeting on foundations of software engineering*. 385–395.

[2] Mahmoud Alfadel, Diego Elias Costa, and Emad Shihab. 2023. Empirical analysis of security vulnerabilities in python packages. *Empirical Software Engineering* 28, 3 (2023), 59.

[3] Sabrina Amft, Sandra Höltervennhoff, Rebecca Panskus, Karola Marky, and Sascha Fahl. 2024. Everyone for Themselves? A Qualitative Study about Individual Security Setups of Open Source Software Contributors. In 45th IEEE Symposium on Security and Privacy, IEEE S&P 2024, May 20-23, 2024. IEEE, IEEE Computer Society.

[4] Nafisa Anjum, Nazmus Sakib, Juanjose Rodriguez-Cardenas, Corey Brookins, Ava Norouzinia, Asia Shavers, Miranda Dominguez, Marie Nassif, and Hossain Shahriar. 2023. Uncovering Software Supply Chains Vulnerability: A Review of Attack Vectors, Stakeholders, and Regulatory Frameworks. In *2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 1816–1821.

[5] Michael Bailey, David Dittrich, Erin Kenneally, and Doug Maughan. 2012. The menlo report. *IEEE Security & Privacy* 10, 2 (2012), 71–75.

[6] Setu Kumar Basak, Lorenzo Neil, Bradley Reaves, and Laurie Williams. 2022. What are the practices for secret management in software artifacts?. In *2022 IEEE Secure Development Conference (SecDev)*. IEEE, 69–76.

[7] Len Bass, Ralph Holz, Paul Rimba, An Binh Tran, and Liming Zhu. 2015. Securing a deployment pipeline. In *2015 IEEE/ACM 3rd International Workshop on Release Engineering*. IEEE, 4–7.

[8] Lujo Bauer, Lorrie Faith Cranor, Robert W Reeder, Michael K Reiter, and Kami Vaniea. 2009. Real life challenges in access-control management. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 899–908.

[9] Giacomo Benedetti, Luca Verderame, and Alessio Merlo. 2022. Automatic security assessment of github actions workflows. In *Proceedings of the 2022 ACM Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*. 37–45.

[10] David Botta, Rodrigo Werlinger, André Gagné, Konstantin Beznosov, Lee Iverson, Sidney Fels, and Brian Fisher. 2007. Towards Understanding IT Security Professionals and Their Tools. In *Proceedings of the 3rd Symposium on Usable Privacy and Security* (Pittsburgh, Pennsylvania, USA) *(SOUPS '07)*. Association for Computing Machinery, New York, NY, USA, 100–111. https://doi.org/10.1145/1280680.1280693

[11] Ramaswamy Chandramouli, Frederick Kautz, and Santiago Torres-Arias. 2024. Strategies for the Integration of Software Supply Chain Security in DevSecOps CI/CD Pipelines.

[12] Cybersecurity and Infrastructure Security Agency (CISA). [n. d.]. Software Bill of Materials (SBOM). https://www.cisa.gov/sbom.

[13] Alexandre Decan, Tom Mens, and Eleni Constantinou. 2018. On the impact of security vulnerabilities in the npm package dependency network. In *Proceedings of the 15th international conference on mining software repositories*. 181–191.

[14] Erik Derr, Sven Bugiel, Sascha Fahl, Yasemin Acar, and Michael Backes. 2017. Keep me updated: An empirical study of third-party library updatability on android. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 2187–2200.

[15] Gabriel Ferreira, Limin Jia, Joshua Sunshine, and Christian Kästner. 2021. Containing malicious package updates in npm with a lightweight permission system. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 1334–1346.

[16] Konstantin Fischer, Ivana Trummová, Phillip Gajland, Yasemin Acar, Sascha Fahl, and Angela Sasse. 2024. On The Challenges of Bringing Cryptography from Papers to Products: Results from an Interview Study with Experts. In *In 33rd USENIX Security Symposium, USENIX Security '24, Philadelphia, PA, USA, August 14-16, 2024*. USENIX Association. https://www.usenix.org/conference/usenixsecurity24/presentation/fischer

[17] Marcel Fourné, Dominik Wermke, William Enck, Sascha Fahl, and Yasemin Acar. 2023. It's like flossing your teeth: On the Importance and Challenges of Reproducible Builds for Software Supply Chain Security. In *In 44th IEEE Symposium on Security and Privacy*.

[18] Andres Freund. 2024. backdoor in upstream xz/liblzma leading to ssh server compromise. https://www.openwall.com/lists/oss-security/2024/03/29/4.

[19] Fabian Niklas Froh, Matías Federico Gobbi, and Johannes Kinder. 2023. Differential Static Analysis for Detecting Malicious Updates to Open Source Packages. In *Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED'23)*. ACM, 41–49.

[20] Betul Gokkaya, Leonardo Aniello, and Basel Halak. 2023. Software supply chain: review of attacks, risk assessment strategies and security controls. *arXiv preprint arXiv:2305.14157* (2023).

[21] Danielle Gonzalez, Thomas Zimmermann, Patrice Godefroid, and Max Schäfer. 2021. Anomalicious: Automated detection of anomalous and potentially malicious commits on github. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 258–267.

[22] Pronnoy Goswami, Saksham Gupta, Zhiyuan Li, Na Meng, and Daphne Yao. 2020. Investigating the reproducibility of npm packages. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 677–681.

[23] Yacong Gu, Lingyun Ying, Huajun Chai, Chu Qia, Haixin Duam, and Xing Gao. 2023. Continuous Intrusion: Characterizing the Security of Continuous Integration Services. In *44th IEEE Symposium on Security and Privacy (S&P'23)*. IEEE.

[24] Yacong Gu, Lingyun Ying, Yingyuan Pu, Xiao Hu, Huajun Chai, Ruimin Wang, Xing Gao, and Haixin Dua. 2023. Investigating Package Related Security Threats in Software Registries. In *44th IEEE Symposium on Security and Privacy (S&P'23)*. IEEE.

[25] Marco Gutfleisch, Jan H. Klemmer, Niklas Busch, Yasemin Acar, M. Angela Sasse, and Sascha Fahl. 2022. How Does Usable Security (Not) End Up in Software Products? Results From a Qualitative Interview Study. In *43rd IEEE Symposium on Security and Privacy, IEEE S&P 2022, May 22-26, 2022*. IEEE Computer Society.

[26] Julie M Haney, Mary Theofanos, Yasemin Acar, and Sandra Spickard Prettyman. 2018. " We make it a big deal in the company": Security Mindsets in Organizations that Develop Cryptographic Products.. In *SOUPS USENIX Security Symposium*. 357–373.

[27] Sandra Höltervennhoff, Philip Klostermeyer, Noah Wöhler, Yasemin Acar, and Sascha Fahl. 2023. "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 *32nd USENIX Security Symposium (USENIX Security 23)*. 2509–2525.

[28] HF Md Jobair, M Tasnim, H Shahriar, M Valero, A Rahman, and F Wu. 2022. Investigating Novel Approaches to Defend Software Supply Chain Attacks. In *33rd IEEE Int. Symp. Softw. Reliab. Eng.*

[29] Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. 2013. Why don't software developers use static analysis tools to find bugs?. In *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 672–681.

[30] Jaap Kabbedijk and Slinger Jansen. 2011. Steering Insight: An Exploration of the Ruby Software Ecosystem. In *Software Business*, Björn Regnell, Inge van de Weerd, and Olga De Troyer (Eds.). Vol. 80. Springer Berlin Heidelberg, Berlin, Heidelberg, 44–55. https://doi.org/10.1007/978-3-642-21544-5_5 Series Title: Lecture Notes in Business Information Processing.

[31] Kelechi G Kalu, Tanya Singla, Chinenye Okafor, Santiago Torres-Arias, and James C Davis. 2024. An Industry Interview Study of Software Signing for Supply Chain Security. *arXiv preprint arXiv:2406.08198* (2024).

[32] Jan H Klemmer, Marco Gutfleisch, Christian Stransky, Yasemin Acar, M Angela Sasse, and Sascha Fahl. 2023. "Make Them Change it Every Week!": A Qualitative Exploration of Online Developer Advice on Usable and Secure Authentication. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2740–2754.

[33] Igibek Koishybayev, Aleksandr Nahapetyan, Raima Zachariah, Siddharth Muralee, Bradley Reaves, Alexandros Kapravelos, and Aravind Machiry. 2022. Characterizing the Security of Github CI Workflows. In *31st USENIX Security Symposium (USENIX Sec'22)*. 2747–2763.

[34] Alexander Krause, Jan H Klemmer, Nicolas Huaman, Dominik Wermke, Yasemin Acar, and Sascha Fahl. 2023. Pushed by Accident: A {Mixed-Methods} Study on Strategies of Handling Secret Information in Source Code Repositories. In *32nd USENIX Security Symposium (USENIX Security 23)*. 2527–2544.

[35] Raula Gaikovina Kula, Daniel M German, Ali Ouni, Takashi Ishio, and Katsuro Inoue. 2018. Do developers update their library dependencies? An empirical study on the impact of security advisories on library migration. *Empirical Software Engineering* 23 (2018), 384–417.

[36] Piergiorgio Ladisa, Henrik Plate, Matias Martinez, and Olivier Barais. 2023. Sok: Taxonomy of attacks on open-source software supply chains. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1509–1526.

[37] Piergiorgio Ladisa, Merve Sahin, Serena Elisa Ponta, Marco Rosa, Matias Martinez, and Olivier Barais. 2023. The Hitchhiker's Guide to Malicious Third-Party Dependencies. In *Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED'23)*. ACM, 65–74.

[38] Chris Lamb and Stefano Zacchiroli. 2021. Reproducible builds: Increasing the integrity of software supply chains. *IEEE Software* 39, 2 (2021), 62–70.

[39] Enrique Larios Vargas, Maurício Aniche, Christoph Treude, Magiel Bruntink, and Georgios Gousios. 2020. Selecting third-party libraries: The practitioners' perspective. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 245–256.

[40] Enrique Larios Vargas, Maurício Aniche, Christoph Treude, Magiel Bruntink, and Georgios Gousios. 2020. Selecting third-party libraries: The practitioners' perspective. In *Proceedings of the 28th ACM joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 245–256.

[41] Elizabeth Lin, Igibek Koishybayev, Trevor Dunlap, William Enck, and Alexandros Kapravelos. 2024. UntrustIDE: Exploiting Weaknesses in VS Code Extensions. In *Proceedings of the ISOC Network and Distributed Systems Symposium (NDSS)*. Internet Society.

[42] Chengwei Liu, Sen Chen, Lingling Fan, Bihuan Chen, Yang Liu, and Xin Peng. 2022. Demystifying the vulnerability propagation and its evolution via dependency trees in the npm ecosystem. In *Proceedings of the 44th International Conference on Software Engineering*. 672–684.

[43] Marcela S Melara and Mic Bowman. 2022. What is Software Supply Chain Security? *arXiv preprint arXiv:2209.04006* (2022).

[44] Courtney Miller, Christian Kästner, and Bogdan Vasilescu. 2023. "We Feel Like We're Winging It:" A Study on Navigating Open-Source Dependency Abandonment. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1281–1293.

[45] Jaron Mink, Hadjer Benkraouda, Limin Yang, Arridhana Ciptadi, Ali Ahmadzadeh, Daniel Votipka, and Gang Wang. 2023. Everybody's got ML, tell me what else you have: Practitioners' perception of ML-based security tools and explanations. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2068–2085.

[46] Jaron Mink, Harjot Kaur, Juliane Schmüser, Sascha Fahl, and Yasemin Acar. 2023. "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 *32nd USENIX Security Symposium (USENIX Security 23)*. 3763–3780.

[47] Siddharth Muralee, Igibek Koishybayev, Aleksandr Nahapetyan, Greg Tystahl, Brad Reaves, Antonio Bianchi, William Enck, Alexandros Kapravelos, and Aravind Machiry. 2023. ARGUS: A Framework for Staged Static Taint Analysis of GitHub Workflows and Actions. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX, 6983–7000.

[48] Shradha Neupane, Grant Holmes, Elizabeth Wyss, Drew Davidson, and Lorenzo De Carli. 2023. Beyond typosquatting: an in-depth look at package confusion. In *32nd USENIX Security Symposium (USENIX Security 23)*. 3439–3456.

[49] Alfred Ng. 2018. US: Russia's NotPetya the Most Destructive Cyberattack Ever. https://www.cnet.com/news/privacy/uk-said-russia-is-behind-destructive-2017-cyberattack-in-ukraine/ Checked 2023-11-10..

[50] Sabato Nocera, Simone Romano, Massimiliano Di Penta, Rita Francese, and Giuseppe Scanniello. 2023. Software bill of materials adoption: a mining study from GitHub. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 39–49.

[51] Marc Ohm, Henrik Plate, Arnold Sykosch, and Michael Meier. 2020. 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, 23–43.

[52] Chinenye Okafor, Taylor R Schorlemmer, Santiago Torres-Arias, and James C Davis. 2022. Sok: Analysis of software supply chain security by establishing secure design properties. In *Proceedings of the 2022 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED'22)*. ACM, 15–24.

[53] OpenSSF. [n. d.]. OpenSSF Scorecard. https://securityscorecards.dev/.

[54] Ivan Pashchenko, Henrik Plate, Serena Elisa Ponta, Antonino Sabetta, and Fabio Massacci. 2018. Vulnerable open source dependencies: Counting those that matter. In *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*. 1–10.

[55] Ivan Pashchenko, Duc-Ly Vu, and Fabio Massacci. 2020. A qualitative study of dependency management and its security implications. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 1513–1531.

[56] Sean Peisert, Bruce Schneier, Hamed Okhravi, Fabio Massacci, Terry Benzel, Carl Landwehr, Mohammad Mannan, Jelena Mirkovic, Atul Prakash, and James Bret Michael. 2021. Perspectives on the solarwinds incident. *IEEE Security & Privacy* 19, 2 (2021), 7–13.

[57] Akond Rahman, Effat Farhana, and Laurie Williams. 2020. The 'as code' activities: Development anti-patterns for infrastructure as code. *Empirical Software Engineering* 25 (2020), 3430–3467.

[58] Akond Rahman, Chris Parnin, and Laurie Williams. 2019. The seven sins: Security smells in infrastructure as code scripts. In *IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 164–175.

[59] Mario Silic and Andrea Back. 2013. Information Security and Open Source Dual Use Security Software: Trust Paradox. In *Open Source Software: Quality Verification*, Etiel Petrinja, Giancarlo Succi, Nabil El Ioini, and Alberto Sillitti (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 194–206.

[60] Vibha Singhal Sinha, Diptikalyan Saha, Pankaj Dhoolia, Rohan Padhye, and Senthil Mani. 2015. Detecting and mitigating secret-key leaks in source code repositories. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 396–400.

[61] Sonatype. 2024. What is a software supply chain? https://www.sonatype.com/resources/software-supply-chain-management-part-1-what-is-a-software-supply-chain

[62] Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. 2024. Boms away! inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–13.

[63] The White House. 2021. Executive Order on America's Supply Chains (EO14017). https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.

[64] The White House. 2021. Executive Order on Improving the Nation's Cybersecurity (EO14028). https://www.whitehouse.gov/briefing-room/presidential-actions/2021/05/12/executive-order-on-improving-the-nations-cybersecurity/.
[65] Tyler W Thomas, Madiha Tabassum, Bill Chu, and Heather Lipford. 2018. Security during application development: An application security expert perspective. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 1–12.
[66] Greg Tystahl, Yasemin Acar, Michel Cukier, William Enck, Christian Kastner, Alexandros Kapravelos, Dominik Wermke, and Laurie Williams. 2024. S3C2 Summit 2024-03: Industry Secure Supply Chain Summit. arXiv:2405.08762 [cs.CR] https://arxiv.org/abs/2405.08762
[67] Marat Valiev, Bogdan Vasilescu, and James Herbsleb. 2018. Ecosystem-level determinants of sustained activity in open-source projects: A case study of the PyPI ecosystem. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 644–655.
[68] Duc-Ly Vu, Fabio Massacci, Ivan Pashchenko, Henrik Plate, and Antonino Sabetta. 2021. Lastpymile: identifying the discrepancy between sources and packages. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 780–792.
[69] Dominik Wermke, Jan H Klemmer, Noah Wöhler, Juliane Schmüser, Harshini Sri Ramulu, Yasemin Acar, and Sascha Fahl. 2023. "Always Contribute Back": A Qualitative Study on Security Challenges of the Open Source Supply Chain. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1545–1560.
[70] Dominik Wermke, Noah Wöhler, Jan H Klemmer, Marcel Fourné, Yasemin Acar, and Sascha Fahl. 2022. Committed to trust: A qualitative study on security & trust in open source software projects. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1880–1896.
[71] Evan D Wolff, KM Growley, MG Gruden, et al. 2021. Navigating the solarwinds supply chain attack. *The Procurement Lawyer* 56, 2 (2021).
[72] Elizabeth Wyss, Lorenzo De Carli, and Drew Davidson. 2023. (Nothing But) Many Eyes Make All Bugs Shallow. In *Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses (SCORED'23)*. ACM, 53–63.
[73] Bowen Xu, Le An, Ferdian Thung, Foutse Khomh, and David Lo. 2019. Why reinventing the wheels? An empirical study on library reuse and re-implementation. *Empirical Software Engineering* 25, 1 (Sept. 2019), 755–789. https://doi.org/10.1007/s10664-019-09771-0
[74] Dapeng Yan, Yuqing Niu, Kui Liu, Zhe Liu, Zhiming Liu, and Tegawendé F Bissyandé. 2021. Estimating the attack surface from residual vulnerabilities in open source software supply chain. In *2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 493–502.
[75] Awad A Younis, Yi Hu, and Ramadan Abdunabi. 2023. Analyzing Software Supply Chain Security Risks in Industrial Control System Protocols: An OpenSSF Scorecard Approach. In *2023 10th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 302–311.
[76] Nusrat Zahan, Parth Kanakiya, Brian Hambleton, Shohanuzzaman Shohan, and Laurie Williams. 2023. Openssf scorecard: On the path toward ecosystem-wide automated security metrics. *IEEE Security & Privacy* 21, 6 (2023), 76–88.
[77] Nusrat Zahan, Elizabeth Lin, Mahzabin Tamanna, William Enck, and Laurie Williams. 2023. Software bills of materials are required. are we there yet? *IEEE Security & Privacy* 21, 2 (2023), 82–88.
[78] Nusrat Zahan, Shohanuzzaman Shohan, Dan Harris, and Laurie Williams. 2023. Do software security practices yield fewer vulnerabilities?. In *2023 IEEE/ACM 45th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 292–303.
[79] Nusrat Zahan, Thomas Zimmermann, Patrice Godefroid, Brendan Murphy, Chandra Maddila, and Laurie Williams. 2022. What are weak links in the npm supply chain?. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*. 331–340.
[80] Markus Zimmermann, Cristian-Alexandru Staicu, Cam Tenny, and Michael Pradel. 2019. Small World with High Risks: A Study of Security Threats in the npm Ecosystem. In *Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19)* (Santa Clara, CA, USA). USENIX, USA, 995–1010.

A Interview Guide

Q1: Introduction

(1) Please tell me a bit about your biography, how did you get into the field of software development?
  (a) What is your experience in software development?
  (b) What industry do you work in?
  (c) Do you attend any conferences, workshops or seminars as part of your professional activities?
(2) Does your employer offer benefits in the form of funding (full or partial) for educational courses?
If yes: (a) Have you taken software security courses as part of this funding?
  (b) Do you consider this experience useful? Has it been helpful in your work? How?
If no: (a) In which field would you take a course if the opportunity were available?

Q2: Awareness of Software Supply Chain and Security Methods

(1) Are you familiar with the concept of software supply chain?
If yes: (a) Can you name key components of a software supply chain? *[Share information and ask b & c]*
  (b) Which components do you use in your daily work?
  (c) In your opinion, which component is the most vulnerable in terms of security? Why?
[Next share the main methods for SSC security and proceed]
(2) Were any of these methods familiar to you?
If yes: (a) Which methods do you use in your everyday work?
  (b) In your opinion, which of these methods are the most challenging to implement or use? Why?
(3) Does your company organize seminars or workshops on security-related topics?
(4) Have you used any sources of information to broaden your knowledge of security? If so, which ones?

Q3: Security Issues and Challenges

(1) How would you rate the security status of the current project you are working on on a scale from 1 to 5, where 1 indicates insecure and 5 indicates highly secure? Do you think there are security vulnerabilities?
(2) What are the main challenges you face in securing your project (e.g.: time, cost, specificity of some tools: difficult to use or need additional training, etc.)?
(3) Are there specific guidelines or standards that you follow to ensure project security?
(4) What recommendations would you give to other developers regarding approaches to ensuring the security of their projects? What would you consider very important in terms of project security?
(5) Can you think of any supply chain related security issues your projects have faced in the past?

Q4: Tools and Methods

(1) What problems do you most often encounter when using security tools? Have you had to change or stop using certain tools or components (e.g.: frequent updates which caused inconveniences, or the project was abandoned, etc.)? If so, why?
  (a) What do you think could be improved or enhanced?
(2) If and when you want to include third-party components into your projects, do you check them for vulnerabilities? If so, how do you do that?
(3) Do you audit the dependencies used in your project?
(4) Are there roles in your project related to security?
(5) Do you keep your knowledge of new security tools and techniques up to date? If yes, how?

Q5: Security in the Development Lifecycle

(1) During which phases of the development process do you place specific emphasis on security?
(2) Do you sometimes have to find a compromise between security concerns and the need for speed of development?
  (a) How do you allocate time for product development and security? What do you prioritize?
(3) Have you had to make changes in the development process to improve security? If yes, please tell us what changes were made?
(4) If you want to use third-party components or tools in your projects, what is the criteria for choosing them, what metrics?
  (a) Do you document all third-party dependencies or tools that are included in the project?
(5) Opinion about having security guidelines within a company/team/project?