ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

"Threat modeling is very formal, it's very technical, and also very hard to do correctly": Investigating Threat Modeling Practices in Open-Source Software Projects

Harjot Kaur[‡*], Carson Powers[†*], Ronald E. Thompson III[†], Sascha Fahl[‡], Daniel Votipka[†]
[‡]*CISPA Helmholtz Center for Information Security;* [†]*Tufts University*

Abstract

Vulnerabilities in open-source software (OSS) projects can potentially impact millions of users and large parts of the software supply chain. Rigorous secure design practices, such as threat modeling (TM), can help identify threats and determine and prioritize mitigations early in the development lifecycle. However, there is limited evidence regarding how OSS developers consider threats and mitigations and whether they use established TM methods.

Our research is the first to fill this gap by investigating OSS developers' TM practices and experiences. Using semi-structured interviews with 25 OSS developers, we explore participants' threat finding and mitigation practices, their challenges and reasons for adopting their practices, as well as desired support for implementing TM in their open-source projects. Because OSS development is often a volunteer effort, decentralized, and lacking security expertise, more structured TM methods introduce additional costs and are perceived as having limited benefit. Instead, we find almost all OSS developers conduct TM practices in an ad hoc manner due to the ease-of-use, flexibility, and low overhead of this approach. Based on our findings, we provide recommendations for the OSS community to better support TM processes in OSS.

1 Introduction

A substantial amount of software depends on open-source software (OSS) projects, which provide commonly used libraries and frameworks. Up to 96% of codebases depend on OSS [86], many of these OSS dependencies have known vulnerabilities, and experts believe OSS security is critical in ensuring the software supply chain's integrity [38, 84, 96]. A recent industry report audited 1,067 commercial codebases, finding 77% of their code originated from OSS, and 84% of those projects introduced at least one vulnerability originating from OSS code [86]. Unmitigated vulnerabilities in OSS dependencies can jeopardize the entire software supply chain and affect millions or billions of users, as happened in the Log4Shell vulnerability [6], which caused widespread panic [21, 40, 87]. To ensure the integrity of the software supply chain, we must improve OSS security.

[*]These authors contributed equally to this work.

Secure Design using Threat Modeling. One possible solution is to improve security practices during code design to achieve "secure by design" software [22, 55]. One approach to secure design is using threat modeling (TM), "[a] process of using hypothetical scenarios, system diagrams, and testing to help secure systems and data" [23]. TM is a practice recommended by many organizations [22, 23, 25, 29, 51, 67, 78]. Tasks associated with TM can be broken into a set of guiding questions developed by experts in the TM community, known as the "Four Questions": *What are we building?*, *What can go wrong?*, *What are we going to do about it?*, and *Did we do a good enough job?* [80, 102]. These questions provide an overarching structure for framing TM research that does not constrain participants [89]. TM emerged as a way to provide more structure to what previously was an ad hoc practice, whereby threats are noted, and possibly mitigated, within a system as they arise or from an arbitrary list of threats (we use "ad hoc" going forward to describe these unstructured practices). Using ad hoc practices likely creates issues with completeness, rationale, and consistency [11, pg. 16-17]. More structured processes have emerged to alleviate these issues, which establish specific steps to use as well as ways to identify, classify, and document threats [55, 80]. These include STRIDE [55], PASTA [90], attack trees [75], and others [30, 78, 88]. We collectively describe these established and documented TM processes as "structured." While there has not been a methodical evaluation of the challenges of TM, anecdotal evidence suggests that TM requires significant expertise [49], requires significant resource coordination [93], and in practice can be exclusionary and opaque [76, pg. 87].

Though some resources suggest how to conduct structured TM in OSS [97], it is generally assumed OSS projects rarely use these processes [26,34]. However, to the best of our knowledge, analysis of whether and how OSS developers consider threats and mitigations during design has not been conducted.

We fill this gap by interviewing 25 OSS contributors and maintainers (collectively referred to as "OSS developers") regarding how they think about threats and mitigations during design and whether they follow ad hoc practices or use some version of structured TM processes. We also investigate the challenges they face conducting TM.

Specifically, we answer the following research questions:

RQ1 [*Process*]: What are OSS projects' practices for identifying and mitigating threats?

RQ2 [*Challenges*]: What are the obstacles and challenges to using threat identification and mitigation practices in OSS projects?

RQ3 [*Adoption*]: What motivated OSS projects to adopt practices for identifying and mitigating threats?

We found almost all OSS developers' practices were best described as ad hoc. That is, TM was not necessarily a regular part of their development process, but threats and mitigations were considered as the developer thought they might be relevant. In many cases, this meant the developer maintained a mental set of common threats or mitigations and went through the checklist if they believed the current feature or asset might warrant scrutiny.

A few participants reported a structured process where TM is a regular step in the software development process. Many of these participants followed established methods such as STRIDE and attack trees, but others used bespoke solutions such as their own checklists of threats common to their software stack. Of note, participants using ad hoc practices report threats independently, e.g., in an issue tracker, but not in a manner which allows a full view of the system threat model.

We also observed several challenges unique to TM in OSS, which motivated ad hoc practice adoption. Most notably, many OSS developers are volunteers and have limited time. This motivated our participants to seek easy to use threat identification and mitigation practices with limited overhead. Similarly, because few developers are security experts, some participants viewed the additional overhead unhelpful believing structured TM processes would not be effective without security expertise. Also, OSS projects are decentralized making it challenging to keep track of a full-system threat model. This motivated participants to seek flexible processes that could be applied to individual segments and shared among several lightly coordinated parties. While participants believed ad hoc practices are not ideal as they will not be applied in all cases, they allow developers to consider program segments in isolation during threat identification and mitigation. Participants viewed structured processes as requiring regular whole-system reviews. While this is not necessarily true, as Shostack has argued structured processes can be adapted to be more lightweight [81], we note most participants view established methods like STRIDE and attack trees as heavy-weight, leading most to develop their own approaches.

Key Contributions.

- Through interviews with 25 OSS developers, we show that almost all used ad hoc TM practices and a few had structured processes integrated into their workflow
- We also identify reported challenges to TM, which include volunteers having limited motivation to do TM, lack of access to security expertise, and the decentralized structure of OSS teams
- We give suggestions for TM process and tool improvements, a future study of ad hoc practices' effectiveness, and how to effectively prioritize TM adoption in OSS

2 Related Work

We discuss related work in three key areas: OSS security practices, developer security behaviors, and TM.

OSS Security Practices. OSS projects provide researchers with readily accessible software and data that can reveal how security challenges manifest in the real world. Much of this work investigates publicly available development artifacts to make conclusions about the prevalence and form of security practices in OSS [15, 35, 44, 50, 57, 69, 77]. For instance, Favato et al. looked at how the number of developers affects the incidence of security flaws, finding projects with more developers are associated with fewer flaws [35]. However, there have been few user studies investigating OSS security practices. Most notably, Wermke et al. investigated security practices and trust within the OSS community [96]. They interviewed OSS developers to learn the security practices OSS projects employ, including challenges to adoption. While projects may have large contributor bases, the number of core contributors is much smaller, and core contributors implement most changes. Additionally, they found practices differed significantly among projects, and many OSS developers believe automation only solves so much. We build on this work by investigating a specific security practice, TM, and understanding what challenges OSS developers face in adoption.

Developer Security Behaviors. Prior work has investigated how developers more generally approach security throughout the development lifecycle. This work has explored how organizations impact developers' approaches to security [13, 14, 41, 48, 68, 96], and how developers approach and implement security [9, 62, 64, 71, 73, 95]. Most relevant to our work is the research focusing on secure design practices. Haney et al. looked at cryptographic developers' security practices and mindsets, finding that a strong security culture is not linked to organization size nor resources available [48]. These findings are relevant to our work as they suggest a strong security culture should be able to exist in OSS, and that a form of TM, whether ad hoc or structured, would be present. Palombo et al. used an ethnographic study to understand how security is implemented in the development lifecycle within a company [68]. Work focusing on developer security practices provides an important view of the software lifecycle; while similar to our work, it does not address the specific issues facing OSS developers nor their TM approaches.

Threat Modeling User Studies. Several studies have looked at TM using user studies [16, 37, 39, 42, 82, 83, 85, 89, 92, 93]. This work can be divided by research focusing on practitioners [16, 37, 39, 82, 83, 85, 89, 93] and those using students in controlled lab settings [42, 92]. Of the work focused on practitioners, some has prescribed specific TM processes to understand the challenges when adopting them [37, 85]. For example, Stevens et al. performed a field observation in New York City Cyber Command, observing participant TM practices over time after training team members on a specific TM process, Center of Gravity analysis [85]. Shull et al. conducted a controlled experiment, assigning participants to use one of three established TM processes and asking them to identify threats in an emulated system [37]. Of the research that has been more exploratory [39, 82, 83, 89, 93], both Thompson et al. and Shreeve et al. leveraged scenario-based research to observe how practitioners approach TM. They found security experts and decision-makers often considered mitigations before the associated risk and that these users relied on flexible approaches [39, 82, 83, 89]. Most similar to our work is Verreydt et al., which looked at TM adoption in Dutch organizations [93]. They found TM adoption faces several challenges, including coordinating stakeholders and documentation needs, having sufficient resources, and systematically following up on the results. Unlike our work, they focused on organizations actively using TM due to compliance requirements in their sectors, and their participants were active advocates for TM adoption. While we believe our results have broader implications for TM across organization types, our work focuses on OSS projects presenting unique organizational challenges and security processes different from previously investigated settings [15, 16, 35, 50, 57, 69, 96].

3 Methodology

This section describes the interview guide development, including input from domain experts. Then, we explain our data collection process, including the screening survey and interview, as outlined in Figure 1. We describe our recruitment strategies, data analysis and the work's limitations.

3.1 Community Engagement

Because there is no prior investigation of TM in open-source communities, we began by engaging with senior OSS community members to understand the community's language and dynamics better. This engagement supported our questions' face validity by ensuring appropriate terminology use. It also provided helpful context to appropriately interpret participant responses.

We contacted organizations mentioned in two lists of OSS organizations. The first was a curated list of organizations displaying their open source projects on `github.com` [43], which suggested they might be open to discussing their TM practices. The second was a curated list of organizations providing support services, e.g., security audits, to OSS projects on `opensource.com` [65]. We expected these organizations might provide useful insights into security practices of the projects they support. We also contacted members of our professional networks with significant OSS experience.

We had informal conversations with four OSS professionals from different projects. Two were recruited from the `github.com` and `opensource.com` lists, and two were through our professional contacts. Of these four, two were sysadmins who host OSS projects and two were OSS developers. These discussions helped us refine research questions and develop a recruitment strategy, so they did not complete the interview and are not pilots. They all had experience supporting OSS projects, either generally, for security, or both. In each conversation, we inquired about the channels OSS developers use to discuss threats, the terminology developers use when talking about threats, and suggestions for recruiting OSS developers in a non-invasive way. Additionally, we asked if they provided OSS projects with resources for secure development and if they knew of OSS projects that conducted structured or ad hoc TM activities.

These conversations taught us that though some OSS projects conduct TM processes, these practices and the resulting models are rarely publicly shared. This motivated us to design an interview study investigating these TM practices, instead of an artifact analysis. These discussions also revealed the terms "threat" and "threat modeling" are not universally used. Also, prior work suggests these definitions can differ [99] and we observed differences among TM resources [23, 25, 29, 61, 67]. To ensure we captured a broad set of perspectives, we chose to use "threats/vulnerabilities/security issues" together (Section 3.4 contains interview content).

3.2 Screening Survey

The first step in our study flow was a short screening survey (Figure 1.A), which obtained informed consent and asked about participants' OSS development experience. Our screening survey is given in our supplemental materials [3].

Questions and Eligibility Criteria. After obtaining informed consent, the remaining survey questions assessed interview eligibility (Figure 1.B). To be interview eligible, participants first needed OSS development experience. This included participants who self-reported at least one year of OSS development experience and regular OSS project contributions. To assess their activity in OSS, we asked participants to provide a link to a public page verifying their contributions.

Our goals were to learn what threat-finding and mitigating processes OSS developers use (RQ1), their reasons for adopting and using TM (RQ3), what challenges exist (RQ2), and what improvements they would like to see. Hence, we

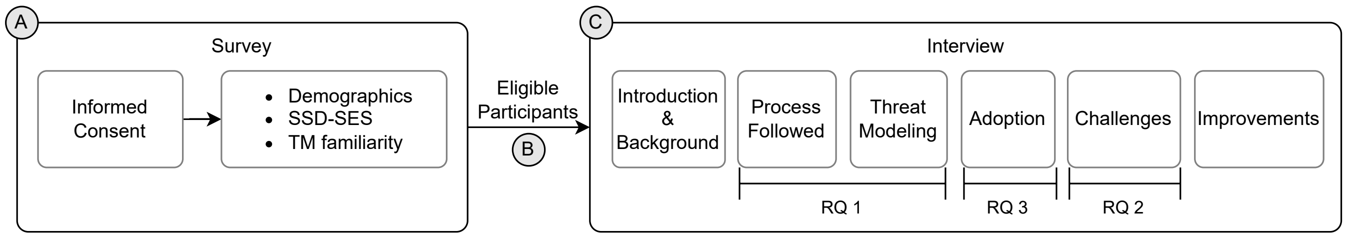

Figure 1: Study flow.

also needed participants who had thought about threats to their projects or at least knew how to think about threats. To assess participants' background, we included seven questions adapted from the Secure Software Development Self-Efficacy Scale (SSD-SES) [94] which asked how frequently participants think about threats and one question asking about familiarity with established TM processes. If the participant indicated they perform threat-finding activities at least "sometimes," they could answer RQ1 and possibly RQ3 questions, so they were eligible. If they agreed or strongly agreed with the statement "I am familiar with at least one threat modeling method (e.g., STRIDE, PASTA, Attack trees, etc.)," they could answer RQ2 questions and questions about TM improvements, so they were eligible. Overall, if they passed the eligibility check and the OSS experience check, they were eligible. We did not require participants to have experience with or knowledge of established TM methods. As long as they reported previously considering security threats at some point during development, they were eligible.

After conducting five pilot interviews, we added these TM background questions, as participants who neither performed, nor knew about threat identification and mitigation processes could not answer our interview questions.

3.3 Interview Procedure

After screening for eligibility, two authors conducted the interviews collaboratively, alternating as the primary interviewer. The primary interviewer conducted the interview (i.e., read the interview script and asked questions), while the supporting interviewer took notes and asked follow-up questions when appropriate. Both interviewers followed a structured and detailed interview guide to ensure consistency. The interview guide also contains general background and instructions, outlining interview goals, stressing the importance of avoiding priming or leading participants, and presenting guidelines for conducting effective interviews based on the original work by Rader et al. [70] and adapted work by Mink et al. [60]. The interview guide is provided in supplemental materials [3] and the interview questions are presented in Appendix A. We compensated interview participants with $40.

Pilot Test. We conducted pilot interviews to ensure the question wording was clear and our questions captured responses relevant to our research questions. We piloted and refined our interview questions with five OSS developers. After five pilots, we decided to split process questions to ask how TM practices differ between design and implementation stages. Since we extracted details about both stages from three of five pilots through follow-up questions, thus receiving answers to all questions in our final interview protocol, we include these three pilots in our final sample of 25 and exclude the remaining two.

3.4 Interview Structure

Interviews consisted of six segments (S1-S6) based on our research questions as illustrated in Figure 1-C. We structured TM process questions around the "Four Questions", which, as discussed in Section 1, provide a high-level structure without constricting participants [89]. To avoid priming, and due to the lack of cohesive definition from our community engagement (Section 3.1) or prior literature [23, 25, 29, 61, 67, 99], we avoided using the term "Threat Modeling" at the beginning of our interviews. Instead, we used the phrases "look for threats/vulnerabilities/security issues" for identification and "solutions to security issues" for mitigation. We introduced our definition of "Threat Modeling" in S3 (see below), which encompasses structured and ad hoc practices. We defined "formal TM" as structured processes (e.g., STRIDE, PASTA, attack trees). We referred to formal TM exclusively when it was needed. All interview questions are listed in Appendix A.

S1: Introduction and Background. At the start of the interview, we introduced ourselves and provided a brief study overview. Next, we asked participants about their OSS experience, their projects, OSS development roles and responsibilities, and details about external contributors, as well as if their projects are maintained or supported by an organization. We also asked their opinion on the importance of security, both in general and concerning their projects.

S2: Process Followed (RQ1). We then asked the participants how they visualize the systems they build (e.g., diagramming), their threat identification and mitigation practices, how they communicate and document threats and mitigations, who is responsible for identifying and mitigating threats, the resources used, and the challenges faced when applying their process. Throughout, we asked whether these practices differed between the design and implementation phases and how they resume when later changes are made to the code. We frequently asked probing questions to ensure we covered all development stages. For example, "How do you look for threats *before you begin writing code*" and "Do your threat brainstorming processes differ once you have begun writing code?"

S3: Threat Modeling (RQ1). Next, we asked if they had heard of TM, whether they use a structured process (e.g., STRIDE), how they defined TM, and whether they believed their process met their definition of TM. To ensure a common understanding when asking subsequent questions, we gave our definition of TM, i.e., "the general process of finding threats to a system, as well as what to do about those threats."

S4: Adoption (RQ3). Then, we covered participants' reasons for using and adopting their process, as well as any advice they would give others looking to adopt a TM process. If they did not use structured TM, we asked why they chose not to use a structured approach. If the participant was part of a team when TM was adopted, we asked how the group embraced their TM process. We adapted adoption questions from Fulton et al.'s investigation of Rust adoption [41].

S5: Challenges (RQ2). In the fifth segment, we asked participants whether they believe their process ensures sufficient security. We then asked about challenges limiting their TM process (if applicable), challenges to TM caused by external contributors, and difficulties getting support when they are unsure how to use a particular TM process.

S6: Improvements. Finally, we asked participants to suggest ways to support TM in OSS projects, including desired resources. We also asked whether they would do anything differently if they used TM in another open-source project.

3.5 Recruitment Strategies

We recruited participants using direct messages to project developers, social media posts, professional contacts, postings on Upwork.com [8], and snowball sampling [45]. Because we aimed to reach saturation in structured and ad hoc TM processes used in OSS projects, we recruited participants from a broad set of project types and sizes.

We prioritized contacting contributors to security-related projects since we assumed security-related projects are more likely to use TM, though contributing to a security-related project was not a requirement. We broadly define "security-related" as: (a) projects in which security incidents could have a broad impact (e.g., large user base), (b) projects for which security *is* the product (e.g., cryptography), (c) projects that may have significant financial risk (e.g., project is supported by a corporation), or (d) projects utilizing tools associated with security considerations (e.g., programmed in Rust).

To find potential participants, we used several lists of projects hosted on GitHub and found email addresses for the top contributors. Since using email addresses found on GitHub violates acceptable use policies [7], we searched for personal or project-related websites with contact information.

To find projects satisfying criteria (a), we searched for projects listed on NPM (`https://www.npmjs.com/`) containing the "security" or "cryptography" keywords, then contacted the most downloaded projects first. To find instances for (b), we extracted GitHub projects with the tag "cryptography." We excluded repositories without source code, e.g., a repository with collections of documents. For (c), we used Google search with the key phrase "end user open source software" and recorded all projects on the first results page. Finally, for (d), we downloaded the database of crates.io [1], a centralized repository for Rust packages. For the cryptography tag (b) and Rust repositories (d), we sorted the lists of repositories by highest star and fork count first, similar to Wermke et al. [96]. For consistency, we removed projects from (c) and (d) that were not hosted on GitHub. To ensure we contacted active projects, we ignored projects without at least one commit in six months prior to the start of data collection.

Also, we recruited our professional contacts with experience developing OSS. We received one participant from Upwork and one from snowball sampling. Due to interface restrictions, we used an abridged screening survey for Upwork (in supplemental material [3]). However, the Upwork participant later completed the remaining demographic questions. We stopped recruiting after reaching thematic saturation [74] at 19 interviews and completed interviews for participants who were already scheduled. We conducted interviews primarily via video-conferencing software, except for one participant who answered questions via email.

3.6 Data Analysis and Coding

We used a GDPR-compliant service to transcribe interview recordings. Due to technical issues, we did not have the recording for one eligible participant and instead coded our detailed notes for that participant. One additional eligible participant answered all interview questions via email.

We used thematic analysis with a mix of deductive and inductive coding to analyze the transcripts [24, 27]. We coded at the interview level, applying one code even if the practice was mentioned multiple times. Two researchers created an initial version of the codebook based on the interview guide which was then refined using comments from all authors. Two researchers then independently coded 16 interviews, allowing additional codes to arise from the data, meeting after each to resolve disagreements and adjust the codebook. The same two researchers then independently coded interviews in three rounds of three interviews each, meeting after each round to calculate inter-rater reliability and resolve disagreements.

They achieved agreement with Krippendorff's $\alpha \geq 0.94$ for each variable, indicating sufficient agreement [56]. The same two researchers collaboratively organized and interpreted the qualitative data using affinity diagramming [17] to identify key themes and patterns. Our full codebook and per-variable Krippendorff's α values are in our supplemental materials [3].

3.7 Limitations

Some of our study's limitations are typical of qualitative research. This encompasses self-reporting, under- and over-reporting, social desirability bias, and recall bias. The framing effect in interview studies can potentially introduce bias in participants' responses. We avoided leading questions and carefully crafted unbiased questions to avoid this effect. To reduce social desirability bias, we assured participants throughout the interview that we were not testing or judging their practices.

We only interviewed developers of active projects and may have missed challenges unique to inactive projects. We do not capture how adding TM may affect project abandonment.

Targeting security-relevant projects for recruitment may introduce sampling bias. Thus, we expect our results are an upper bound as projects with fewer security considerations are less likely to employ TM processes. Because we advertised our study's topic as how OSS contributors "think about threats to their projects," this may introduce self-selection bias. However, we do not make claims about TM practice prevalence among all OSS projects. Though we reached saturation, it is possible we did not capture all TM activities, challenges, and desired improvements. While we included participants who did not have experience using established TM methods, we excluded participants who self-reported never considering potential security threats in any way, i.e., formally or informally. We chose to exclude these developers as they would not have relevant experiences to respond to our questions about their process looking for threats or considering adopting TM methods. However, there are possible challenges and method improvements specific to this population that does not consider security threats during development that we do not include. We should consider our results a first step toward understanding and supporting TM practices in OSS development, guiding future research.

4 Participant Demographics

We interviewed 28 participants. We discarded two responses after determining they gave misleading survey answers and were ineligible, and one due to incomplete data, leaving 25 valid participants. Table 1 includes participant demographics.

We recruited 22 by emailing OSS developers from the npm list, cryptography list, end-user OSS software search, and Rust list. We recruited one participant from Upwork, one from our professional contacts, and one from snowball sampling. The mean interview length was 1 hour and 4 minutes.

Our participants had 12.9 years of OSS development experience on average (median=11), and all have contributed to multiple projects. Participants often held different roles for different projects, including contributor (N=15), maintainer (N=14), security engineer/analyst (N=5), and tech lead (N=3). Some indicated they serve as project leaders in addition to one or more roles mentioned above. The majority of participants contributed to a project that is backed by a company.

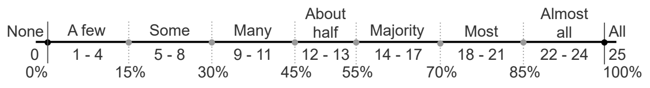

Figure 2: Quantifier terminology used to report the percentage of participants who expressed a given theme.

All participants were involved in at least one security-related project (as defined in Section 3.5). Participants mentioned several reasons for caring about security, such as operating on sensitive data, providing a security product, or feeling a moral obligation. All participants discussed performing structured TM or ad hoc practices in at least one of their projects, though more than half reported at least one project where they did not conduct any form of TM.

We include specific characteristics of participants for context only. While we point to possible relationships between these characteristics and TM practices observed through our thematic analysis, we do not claim these are generalizable. We offer them as possible hypotheses to test in future work.

5 TM Practices in OSS (RQ1)

In this section, we detail participants' different ad hoc practices and structured processes. We discuss practices from only the projects listed in Table 1. Almost all participants used ad hoc threat identification and mitigation practices (which we call "ad hoc practices"), and a few currently or previously used structured TM processes. Of the participants who used ad hoc practices, some did not perceive these as "TM" though they may fit some definitions of TM. We discuss differences in company backing when we see them. We did not see differences between contributor and maintainer roles, which may be explained by many participants serving both roles for different projects. We also describe who participants believe is responsible for TM, how participants communicate, and resources they use. We do not evaluate efficacy of practices, hence, we do not make claims about effectiveness or thoroughness.

We combine or omit specific interview questions or sections for conciseness while broadly following our interview guide. Because this is a qualitative interview study, we do not report exact participant response counts. Instead, we use the quantifier terminology in Figure 2 to refer to theme prevalence, similar to prior works [33, 46, 54, 91, 101].

5.1 System Representation

We began by asking participants how they represent their system when identifying threats. This answers the first key TM question, "*What are we working on?*"

The majority of participants diagrammed their system. These participants used diagrams such as data flow [58], UML

	Interview		Interviewee						Projects			
					Roles							
ID	Interview Duration[1]	Recruitment Channel[2]	TM Experience[3]	Years OSS Experience	Maintainer	Contributor	Security eng.	Tech lead	Project Categories[4]	Company backed[5]	Commits[6]	Contributors[7]
P1	01:06:16	Personal website	●	15	x	-	-	-	Operating system	No	100+	<10
P2	01:14:14	Personal website	○	11	x	-	-	-	Security library, messaging app	Yes	10,000+	100+
P3	00:54:09	Project website	○	20	-	x	x	-	Web browser	Yes	*	*
P4	01:04:21	Personal website	○	10	-	x	x	x	XDR (Extended Detection and Response) system	Yes	100+	10+
P5	01:08:46	Personal website	○	11	-	x	-	-	Encryption tool	Yes	1000+	10+
P6	01:01:12	Personal website	●	45	-	x	-	-	Operating system, messaging app	No	1000+	10+
P7	01:16:56	Personal website	●	10	x	-	-	-	Encryption tool	No	1000+	100+
P8	00:54:43	Personal website	○	10	x	-	-	-	Security library	No	100+	10+
P9	01:00:00	Personal website	○	15	-	x	-	-	Security library, Operating system	Yes	1000+	100+
P10	00:35:48	Personal website	○	24	-	x	-	-	DevOps tooling	No	1000+	100+
P11	01:14:59	Personal website	○	1	x	-	-	-	Cryptocurrency, frontend, operating system, streaming app	Yes	1000+	10+
P12	00:53:20	Personal website	○	13	x	x	-	-	Cryptocurrency	Yes	1000+	100+
P13	00:36:43	Personal website	○	12	x	x	-	-	System software, database	No	100+	10+
P14	00:32:46	Upwork	○	10	x	-	-	-	Security library, password manager	No	10+	<10
P15	01:17:31	Personal website	●	20	-	x	x	-	Security lib, XDR, operating system	Yes	1000+	100+
P16	01:06:03	Personal website	○	11	-	x	-	x	System software, web browser	Yes	100,000+	1000+
P17	-	Personal website	○	7	x	x	-	-	Security library	No	100+	10+
P18	01:36:35	Personal website	○	5	x	x	-	-	Covid app, security library	No	1000+	100+
P19	00:43:08	Snowball	○	8	-	-	x	-	Cryptocurrency	Yes	1000+	100+
P20	01:28:55	Personal website	●	10	x	x	-	-	Cryptocurrency, Security library	Yes	10,000+	100+
P21	01:26:01	Personal website	○	8	-	x	-	-	Security library	Yes	100+	10+
P22	01:14:45	Professional contact	●	16	x	-	x	-	Security library	No	100+	< 10
P23	00:52:03	Personal website	●	4	x	-	-	-	Security library, encryption tool	No	100+	<10
P24	01:23:24	Personal website	●	6	x	-	-	x	Security library, password manager	Yes	1000+	100+
P25	-	Personal website	●	20	-	x	-	-	Password manager	Yes	1000+	10+

[1] Duration unavailable for participant who responded by email and participant without recording. [2] "Personal website" includes websites self-hosted by the participant or an organization for which they work. [3] ● has experience applying formal TM (e.g., STRIDE) in this or other project(s). ○ no previous experience applying formal TM. [4] Covers only projects with practices discussed in the interview. [5] (Yes) if at least one project is company-backed, (No) if none is company-backed. [6] If multiple projects, the number is for the project with the most commits. [7] If multiple projects, the number is for the project with the largest contributors. * We omit data for this participant to protect the privacy of their project.

Table 1: Overview of participant demographics and project metadata. We report binned project metrics to preserve the privacy of our participants and their projects.

class [19], architecture [5], and network diagrams [59] to represent their system structure. Some participants used text-based specifications, including system requirements, functionality, and constraint descriptions of varying detail. Though company-backed projects more often used diagramming, it was not exclusive to this group. All participants who used a structured process used a diagram or written specification.

The remaining participants used mental models. Mental models are internal representations reflecting an individual's understanding of how things work [63]. While everyone uses mental models during TM, participants who do not use diagramming are relying on mental models exclusively. Mental models do not provide shareable and interrogateable externalized views. They are internal maps used to probe system behaviors and threats [72]. A few participants who used mental models mentioned diagramming would not add value. As P15 said, "*I mentally model things. I'm pretty good at holding complex systems in my head. Every time I try to diagram things, just something tends to be lost in the process.*"

Key Takeaways: System Representation.

- The majority of participants diagrammed their system
- Diagramming was slightly more common among company-backed projects

5.2 Threat Identification and Mitigation

Next, we turn to the question "*What could go wrong?*", i.e., threat identification and "*What are we going to do about it*", i.e., mitigation. We consider these steps together as the mitigations often flowed directly from identified threats. We explicitly state below when this is not the case.

Some participants described a background adversarial mindset while writing code. An adversarial mindset, or adversarial thinking, is commonly called the ability to "think like a hacker" [47] or think about how one could subvert the system's rules [28]. This viewpoint is helpful when eliciting threats, but some participants discussed ad hoc practices more as a general adversarial mindset they maintain at all times: "*threat modeling is very much the constant background thought of, hmm, maybe what I'm doing right now is something that somebody could attack, and then stopping and thinking about how would I attack this*" (P1). P18 found this adversarial mindset the natural way to think through threats: "*We never said to ourselves, 'Okay, let's take like two days and just think through the security implications of it.' No, it wasn't like that. It was more organic.*"

Some participants focused on memorized threats. While adversarial thinking can be considered actively brainstorming attacks, other participants reported a more passive ad hoc process of "just knowing" the possible threats in their software domain and where those threats apply. As P12 explained, "*I don't actively look [for threats]. It's mostly common sense.*"

This strategy may work for certain domains. For example, multiple participants developing cryptographic libraries mentioned the set of threats to their domain are well known: "*The class of bugs that you have or threats that you have... is somewhat well-understood for these kind of things*" (P9).

Some participants begin by considering mitigations. These participants' ad hoc practices began by applying mitigations (e.g., use secure tools/frameworks/languages) to their software. As P14 explained, "*I tend to look at [security issues] more from the defensive side, not classifying by threat, but by the defense, by the best practice I use to avoid any potential threats.*" A few participants later brainstormed threats *after* applying mitigations, which supports Thompson et al.'s observation of a similar behavior with medical device threat modeling [89]. However, a few others did not attempt to identify threats. These participants applied secure coding practices and other common mitigations without ever determining if there was an applicable threat requiring the mitigation.

Customized threat and mitigation checklist. One participant who contributes to a large company-backed project mentioned a checklist written by the company's security team used by all developers during code review. This checklist was a product of structured TM with external auditors. P3 reported "*vulnerabilities [are] reported to us, we will often see if we can update the checklist.*" Though utilizing the checklist is not TM *per se*, it is a lightweight application of the threat model. We consider it an ad hoc practice, as contributors need not be familiar with the threat model in order to use it. Though mitigation checklists appear elsewhere [4, 12, 36], this participant's security team tailored it to their product and workflow.

A few participants had structured TM processes integrated into their workflow. One uses STRIDE, and the other attack trees. The STRIDE participant completed a full system analysis after implementing the project and frequently references the threat model documents when changing security-related code. The attack trees participant also refers back to the completed model when making changes to security-related code. As P7 noted: "*I would definitely [revisit the TM] depending on the nature of the changes... it was very helpful to have the threat model and we could then revisit that.*"

An additional participant used LINDDUN when designing a project with privacy concerns, though the project was abandoned before implementation. They preferred ad hoc practices for other projects without these privacy considerations.

Key Takeaways: Threat Identification and Mitigation.

- A few participants used structured processes
- Some considered mitigations before identifying threats
- Some maintained a background adversarial mindset instead of including threat identification as a separate activity

5.3 Responsibility

As OSS projects typically lack the clearly defined roles and responsibilities seen in organizations, we asked participants who is responsible for threat identification and mitigation. They reported several groups, including "all developers," "maintainers," and "the project leader." For company-backed projects, we also saw "security team," "superiors," and "project committee." Specific to finding mitigations, we saw one participant mention the person who *found* the threat is responsible (P15), and another said the person who *introduced* the threat is responsible (P3). In general, we did not see patterns between parties responsible and project type, size, or process/practices used. However, we note two interesting themes below.

Some externalized responsibility for security. When asked who decides their project has met its security goals, some participants placed responsibility outside the project team in two ways: 1) holding users responsible, e.g., "*I hope the people using my code*" (P22), or 2) saying no one is responsible.

OSS development's cornerstones are collaboration, transparency, and active participation from a community of contributors and users [66]. As such, OSS projects tend to receive more community feedback than proprietary projects. We found a few participants rely on community feedback for their design processes, while a few others mentioned users have some responsibility for finding and reporting threats after product release. As P9 stated, "*we rely on people... somebody working on this area posts the design on the mailing list,* *and they rely also on others [developers] critically reading through the design and giving their feedback.*"

However, P15 disagreed strongly with externalizing security responsibility, explicitly stating it is harmful, because users do not have a deep enough system understanding to find threats. They explained, "*You can't expect people using your software to understand the security implications of the software themselves, to do the threat modeling themselves. It's your [the developer's] responsibility to do that.*"

Participants took responsibility for their personal projects. Participants mentioned they take more responsibility for their personal projects than projects with a larger number of core contributors. As P21 noted, the responsible party "*varies across projects that I contribute to. For projects that I own, it's just the code owner.*" Additionally, P14 mentioned "*on my personal projects, I decide. Who else? In some other places where I worked on bigger projects, it would be a security team.*" Perhaps this is an artifact of interviewing participants interested in security, but it seems our participants were less likely to externalize finding threats and mitigations to others when they own the project. This is particularly important, given some small, personally owned projects can become crucial components of the software supply chain.

Key Takeaways: Responsibility.

- Some participants placed responsibility for security outside the project team
- Participants took responsibility for the projects they own

5.4 Communication Practices

As OSS projects rely on collaboration and volunteers, efficient and timely communication is important. It is especially helpful to share threat models so other contributors can ensure their commits are consistent with the planned approach.

About half of participants primarily communicated about identified threats via issue trackers. These issue trackers (e.g., GitHub Issues) allow the public to view identified threats and mitigations (or accepted risk). Using issue trackers is an easy way to document and communicate, because it is already an integral OSS development tool for "*a very GitHub-centric person*" (P21). However, it is difficult to get a complete view of the threat model from these discrete items.

Some participants keep threat information private. When asked how they communicate identified threats during design, some participants mentioned only private communication. A developer who had experience making security contributions to multiple OSS projects noted they rarely can view the project's early security decisions: "*I don't think I've ever really seen a project that goes deeply into the internal security decisions... [it's] maybe discussed in the code comments, but it's otherwise just invisible*" (P15). Another participant shared none of their threat documentation, saying it's "*only on my hard drive*" (P12). The participant who uses attack trees did not share process documentation, nor a list of identified attacks with others. Even the checklist participant (P3) preferred to keep details of the checklist private. This supports our finding in community engagement; some TM practices are not publicly viewable. This practice may be problematic if external contributors are unaware of private design decisions and make incorrect security assumptions when adding code.

Key Takeaways: Communication Practices.

- About half of the participants primarily communicated about threats they identified via issue trackers, which does not show a complete view of the threat model
- Some participants' TM practices were private

5.5 Resources

Almost all participants relied on expected, standard sources for threat identification and mitigation (i.e., books and websites for best practices, research papers, forums, and blogs). We found only company-backed projects relied on security experts. We elaborate on two other noteworthy themes below.

A few participants use ChatGPT to identify threats or mitigations. Just as LLMs have become a regular part of understanding tasks in other computer security domains [100], we observed the same with TM. P20 indicated they use ChatGPT as their only resource for threat identification during design and while coding. P11 uses ChatGPT just for finding mitigations (among other resources): "*I would definitely propose to ChatGPT, for example, 'could you please tell me any possible mitigations for that [vulnerability]?'*"

A few participants relied exclusively on prior knowledge and experiences for threat identification. Prior knowledge or past experiences help shape an individual's thinking pattern and problem-solving approach when facing complex novel challenges [53]. Though we expect everyone uses prior knowledge to some degree while finding threats, a few participants said they rely on prior knowledge exclusively. Though referencing a list of threats may be sufficient if it is community-generated (i.e., a threat library), relying only on what threats a single individual has seen before may lead to missed threats.

Key Takeaways: Resources.

- Only company-backed projects relied on security experts
- A few relied exclusively on prior knowledge and experiences to identify threats
- A few used ChatGPT to identify or mitigate threats

6 Challenges to Using TM in OSS (RQ2)

We observed several common OSS attributes that present unique challenges for TM. In this section, we describe each attribute in turn and how it creates challenges for developers when identifying threats and selecting mitigations.

6.1 OSS Contributors Are Volunteers

The first crucial OSS attribute is that the OSS community is primarily composed of volunteers donating their free time out of passion or personal interest. These contributors may be easily discouraged if forced to spend a portion of their time on overhead. Below, we describe challenges related to this volunteer workforce.

Volunteer contributors lack time and motivation for TM. A few participants mentioned a perceived lack of time for TM. Many OSS developers create software projects as a hobby or in their free time and do not feel they have the time to do TM. This theme appeared in threat identification, mitigation, and overall challenges to adopting TM. P19 mentioned "*I would tell them not to bother with any of those tools [like] STRIDE and whatnot. I don't think they're worth the investment [or] the time.*" While some experts suggest structured TM can be done quickly [81], our participants believed otherwise.

Project owners worry TM requirements will reduce volunteer enthusiasm. Some participants reported other developers – or they themselves – have no interest in the additional steps necessary for structured TM. P13 indicated that this was because structured TM is not fun saying, "*[TM] doesn't sound that much fun, and there's other more fun things to build.*" P10 mentioned adding a meticulous TM process may kill enthusiasm saying, "*Whenever there is too much structure... not so many people are eager to do it... I think establishing a formal process would give quite a lot of pushback and people not wanting to do it.*" Maintaining enthusiasm is already an ongoing concern for these volunteer-fueled projects [20], and adding more overhead threatens momentum.

Key Takeaways: OSS Contributors Are Volunteers.

- Volunteer contributors lack motivation and time for TM
- Overhead for structured TM may hinder momentum

6.2 Decentralized and Transient OSS Teams

OSS projects are often decentralized and lack a hierarchy to facilitate progress. Contributors come and go and may contribute irregularly or inconsistently.

A few participants found it difficult to organize a TM process due to time or geographical barriers. OSS projects tend to have globally distributed contributors. TM is believed to be best when conducted together as a team [102], but due to OSS structure, it becomes hard to gather everyone together to do TM: "*In my experience, [TM] works best when all the people that are owning a component of the system are able to work together or do some whiteboarding together or brainstorming in the same meeting*" (P24). Additionally, the transient nature of OSS teams makes it difficult to rely on a consistent set of contributors to do the TM process. P24 noted one contributor "*did a lot of work on trying to list all operational security threats and issues and mitigations, but it was really like an individual effort and he did that on his own... and also the team disappeared.*"

OSS's decentralized structure complicates determining threat model soundness and completeness. Knowing whether your threat model is sound and complete is a challenge in any setting [18, pg. 10], but the decentralized nature of OSS makes this especially difficult. There is not always a central authority that dictates how the software will evolve. Some contributors then add functionality that the original designers never intended to be part of the project. This new functionality may then introduce threats that were never considered in the original threat model. For example, P24 explained, "*when you are working on open-source stuff with external contributors adding features... you actually never really intended on leaving that, or you really never realized it was part of the system now. You never really considered it in your threat model initially.*" Drift in functionality adds an extra challenge to keeping the threat model up-to-date.

Another participant mentioned they serve as the "shepherd" for their project, who does not actively participate in development, but serves as a gatekeeper when contributors propose new functionality. They noted the difficulty keeping the threat model up-to-date when they are unaware how new functionality was designed. A few participants additionally mentioned external contributors may not be "paranoid" enough to elicit all threats to a system. This is a problem if a project relies on developers maintaining an adversarial mindset, as described in Section 5.2, as they cannot assume all contributors, especially first-time contributors, share this mindset.

A few mentioned consensus in TM can be hard to reach. OSS projects often require consensus from multiple parties for decision-making due to their flat organizational structure. However, there is limited consensus in the security community on the "right" process for TM [78]. This creates a barrier in OSS as parties might disagree whether to adopt TM, what process to adopt, whether an elicited threat applies, and whether the threat warrants mitigation. As P15 stated, "*what I find challenging sometimes is figuring out how to convince other people that the threat exists and needs to be taken seriously, [which is] especially an issue with open-source projects.*" P16 elaborated when discussing reaching consensus for mitigation actions: "*If there is contention about, is this issue worth solving...it usually takes a really long time to make that kind of decision, especially if the answer is no.*"

Key Takeaways: Decentralized and Transient OSS Teams.

- Decentralized structure and transient OSS teams make organizing the TM process and reaching consensus difficult, and complicates evaluating the threat model's soundness and completeness

6.3 OSS Teams May Be Small

While large teams can create issues, a few participants discussed teams too *small* for structured TM to be valuable.

A few participants believed their teams were too small for structured TM to make sense. These participants believed their projects, which have small teams or where they are the only maintainer, would not therefore benefit from TM. P25 mentioned TM is important for large teams only, explaining they have not done structured TM in their OSS project because "*it's just too small of a software and team for that. I've used [structured TM] in non-OSS places, when dealing with Finance/Banking software and teams in the hundreds.*" P11 believed TM would take less time if they had others to work together: "*I would probably stick to an informal process myself just because of the time constraints... you're only alone, you already have enough to build on yourself.*" The participant who did STRIDE did so alone, which for them made the process more boring. Their advice for projects adopting TM was to "*try to involve multiple people with it, because that's going to make it more fun*" (P7).

> **Key Takeaways: OSS Teams May Be Small.**
>
> - A few participants believed structured TM in small teams is too time consuming or lacks benefit

6.4 OSS Projects Lack Security Expertise

Though most company-backed projects had security specialists, other projects must make do without professional help.

Lack of access to security professionals limits threat identification and threat information dissemination. A few participants mentioned the lack of security teams as a challenge to finding threats since they did not have the knowledge necessary to know what to look for. P21 explained, "*You have to have like knowledge and experience.*" Security teams also help disseminate threat information to keep developers up-to-date, as noted by P12: "*they [the security team] share knowledge... there's a channel for InfoSec news. The notion of the team has a lot of information on security like best practices.*" This is a crucial task, as a few participants mentioned they struggle to keep up-to-date with the latest attacks, threats, or best practices. Though having up-to-date knowledge is a challenge for any developer, OSS developers are at a disadvantage for lack of security professionals as a resource.

> **Key Takeaways: OSS Projects Lack Security Expertise.**
>
> - The lack of access to security professionals makes finding threats and disseminating threat information challenging

6.5 Projects Start without Security Concerns

Since some projects begin as hobbyist musings, they lack security concerns at inception. However, security becomes a concern for the fraction of projects that gain widespread use.

A few participants only add security when the project transitions from hobby to use by others. A few developers are not motivated to add any security until they know the project will be used by others. As P13 explained, "*I'm just trying to build something to see if it works... I think it's more important to have a product that people use first, because there's no point building the most secure product if nobody uses it.*" It is arguably unreasonable to expect all developers to add security to their hobby projects they only work on for leisure. However, OSS users should be aware that some products added security only post hoc.

> **Key Takeaways: Projects start without security concerns.**
>
> - Some developers do not consider security until the project is used by others

7 Reasons for Adopting TM (RQ3)

For each challenge identified in Section 6, we describe how this motivates developers' adoption of the processes/practices described previously in Section 5, along with other reasons for TM process adoption beyond these specific challenges.

We consider reasons for adopting and reasons for continuing to use TM practices together, since about half of participants never made a conscious decision to adopt. Instead, their TM practices arose naturally. Overall, the unique attributes of OSS development led participants to strongly consider TM process usability (e.g., ease-of-use, flexibility, and efficiency) when considering adoption, often leading them to ad hoc practices. They also discussed that while a more structured review might promise better security outcomes, they believe the actual gains do not outweigh the extra cost.

7.1 Contributors chose practices that limit overhead and documentation

To address concerns about volunteer contributors' time and motivation (Section 6.1) and challenges posed by small teams (Section 6.3), many participants chose TM practices with little overhead, eliding requirements for detailed documentation.

Some participants focused on time saved using ad hoc practices. These participants believed using a structured process is too time-consuming, instead, choosing ad hoc practices that are "*a bit less time-consuming*" (P24). P3 believed established structured methodologies like STRIDE may be too heavy for smaller open-source projects, as they believe these methods require significant time and record-keeping. Importantly, lack of usability has negatively affected developers' adoption of other security best practices in the past [52, 98]. However, structured TM methods such as STRIDE or attack trees can be applied in a lightweight manner, e.g., conducting a STRIDE analysis on a subset of system components instead of the entire system. Therefore, we are not suggesting our participants' ad hoc practices are more usable. Instead, we highlight these as important properties participants consider.

Participants chose practices that limit creating and maintaining documents. A few participants avoided structured processes to avoid the overhead of creating documents. Since any model produced would be prone to rotting as the code quickly changes, one must continually invest time to keep the model up-to-date. They believed creating documents only begets maintenance with no guarantee that developers will use them. P24 explained: "*it's relatively time-consuming to set up and then it's prone to rotting.*" P1 said "*It's a huge discouraging thing to have to maintain this huge body of text that isn't somehow technically enforceable and hope that people are looking at it, including other maintainers.*"

Some structured processes have low overhead. Though participants believed structured processes strictly require meticulous documentation, it is possible to adapt established structured processes to reduce documents created. A few of our participants mentioned their structured processes are lightweight. P22, who uses attack trees, adopted the process because they could use it without the optional overhead of meticulous documentation. They believe their process incorporating attack trees is not "completely" doing TM, because they do not record all considered risks, which they believed was only necessary if making legal guarantees. Because they still wanted to perform a thorough security review, they adopted this established process, but ignored the suggested documentation steps to limit overhead.

Alternatively, P3, who works on a large company-backed project, described a process where only a few people—the security team—performed structured TM, then developed an easy-to-use checklist for other developers to follow based on the threat model. This reduces the overhead for most developers who can quickly consider security without having to generate their own ideas.

> **Key Takeaways: Choosing practices with limited overhead.**
>
> - Structured processes were believed to be time-consuming and requiring the overhead of documentation
> - Some structured TM can be used in a lightweight manner

7.2 Ad hoc practices are flexible

Since some established structured processes may not be optimal for decentralized teams that lack a strong hierarchy (see Section 6.2), a few developers rejected established processes for a more flexible ad hoc approach. One participant highlighted using a written specification in their process which they keep updated throughout design and implementation. They find this process more flexible and easier to update as the threat model is not a "*set-in-stone*" document (P23). P25 mentioned that since they previously hired auditors for formal security assessments, using a flexible ad hoc approach where they reconsidered issues raised in the prior audit was apt. Using this approach, they can focus on different parts of the system as needed instead of completing a broad system review and repeating effort.

> **Key Takeaways: Ad hoc practices are flexible.**
>
> - A few developers preferred the flexibility of ad hoc practices

7.3 Filling the gap of security expertise

Lack of security expertise (see Section 6.4) led some participants to believe adding TM would provide no benefit. Without expertise, some sought whichever process had the most available support materials.

Without security expert help, a few participants believed structured processes are no better or possibly worse than ad hoc practices. These participants believe structured processes require security expertise to be effective. Without security experts, adopting structured processes would add overhead without benefit. P1 noted structured approaches do not fix the issue of extensive prerequisite knowledge required for effective TM: "*When it comes to threat modeling... it requires a lot of a priori knowledge that you might just not have. Writing cool code and putting it out on GitHub is way easier than having 15, 20 years of experience and understanding what a poison null byte attack in PHP was.*" Structured processes do not help the user understand the threats elicited.

Developers who believe TM can be a stand-in for security expertise may be putting their project at risk by maintaining a false sense of security. P6 reported avoiding existing TM processes like "*design methodology or a threat analysis risk framework*" or "*acronym checklists*" (i.e., STRIDE, PASTA, etc.) for this reason: "*I've seen people using those [frameworks] to fool themselves into thinking they're secure because they checked off all the boxes on their framework... it's probably good to also get an expert to come in from the side and just critique it.*" Some experts acknowledge this tradeoff when advocating for TM adoption among non-experts, and having developers threat model in the absence of security expertise is better than claiming all TM is harmful [80].

Structured TM adopted based on support materials. Availability of support materials drove P7 to adopt STRIDE, which had "*the most documentation, and all public documentation. It was easily available, readily available.*" This helped them understand their project's limitations and risk.

> **Key Takeaways: Filling the gap of security expertise.**
>
> - Structured TM was believed to be beneficial when complemented with security expertise
> - Support material availability drove structured TM adoption

7.4 Other Adoption Decisions

In addition to adoption decisions driven by the unique characteristics of OSS development, we observed other themes in participants' reasoning for choosing TM practices.

Ad hoc practices are sufficient for about half of the participants. These participants knew their approach might not cover *all* threats, but felt confident their ad hoc practices suffice. For example, P19, who worked in high-risk financial environments, was confident maintaining a general adversarial mindset (see Section 5.2) provides sufficient security, and they "*don't need to map out formally our threat models because... it's really inherent to the work that we do.*"

Old, ingrained ad hoc practices resist change. A few participants have entrenched ad hoc practices and are not compelled to transition to a structured process. For example, P4's process was in place before they were aware of structured TM frameworks and noted the deterrent to adopting an structured process is probably a "*symptom of ossification where, 'Oh, we've done it this way, and we seem to work out for us.'*" Since their team members mostly have security backgrounds, they have "*been able to manage in an ad hoc, informal way.*" However, it is possible they may see less benefit from a structured process, which disproportionately benefits projects *without* security-experienced contributors.

A few participants inherited a threat model from a forked project. These participants "inherited" threat models when they forked projects that already conducted structured TM. One assumed the inherited model was sound: "*the [code] which I forked for my repositories has already undergone this [TM] process*" (P11). Another participant reviewed their inherited model and added threats they considered relevant: "*I think, in general, [the parent project] is safe from that attack, but I do plan to make a change*" (P20). They ensured code changes would not invalidate the inherited threat model.

> **Key Takeaways: Other Adoption Decisions.**
>
> - Ad hoc practices were believed to be sufficient even if they do not cover all threats
> - A few projects inherited threat models from projects they forked that had implemented structured TM

8 Discussion

Our results show OSS developers often adopt ad hoc threat identification and mitigation practices. This aligns with Vereydt et al.'s recent investigation of TM in corporate settings [93] and existing general assumptions [26,34]. However, we show ad hoc practices are particularly prevalent due to unique challenges in OSS.

Additionally, we note a general perception among participants that established TM processes like STRIDE and attack trees are inherently structured and heavy weight: "*Things like STRIDE are interesting, but... for the average open source project... those methodologies are too heavy... it would be useful to have other examples of approaches to TM that are still considered TM, but maybe don't have these formalisms attached*" (P3). However, this is not necessarily the case. For example, one of STRIDE's pioneers has argued for the mnemonic to be used in a quick, ad hoc manner to support more systematic brainstorming [81]. This misconception potentially limits the use of existing processes that could improve practices. For example, our participants using mental checklists could incorporate the STRIDE and LINDDUN threat categories into their assessments. Conversely, even though our participants completed most TM process steps (i.e., system diagramming, threat identification, and mitigation), because they did not follow a structured TM process, they did not believe they performed any TM (see Section 5). These perceptions suggest a potential form of functional fixedness [10, 32] around TM processes among our participants which could inhibit adoption of even minimal structure.

Participants' actions and perceptions motivate two categories of recommendations: 1) TM process and tool development that more prominently considers usability and presents how processes can support ad hoc practices, and 2) future research empirically assessing the efficacy of the ad hoc practices organizations use. We conclude with recommendations in both categories and recommendations to OSS leaders and stakeholders for improved OSS security. We note because our recommendations are based on a small sample of developers' responses, they require future validation. However, they provide directions for possible improvement.

8.1 Recommendations for Structured TM Process and Tool Improvement

TM processes and tools should consider ease-of-use, flexibility, and efficiency first-order priorities. Part-time volunteer contributors are unlikely to use any TM process unless it is highly lightweight and flexible. As indicated, this is already true of many existing structured TM processes. It is possible participants believed otherwise because tooling and documentation of these processes either do not support these lighter weight versions or our participants were unaware they exist. One step toward resolving this issue is for future TM tools to support GitHub integration (specifically requested by a few participants), a framework commonly used in OSS.

Similarly, a few participants wanted additional documentation support. Established methods, such as STRIDE and PASTA, have some guidance for how they should be used, but they lack clear examples of application and discussion of how they can be used informally. Therefore, the development of templates and simple example models could be beneficial. For example, P15 wanted a step-by-step guide of proper procedures: "*a runbook-type document for a specific process with templates. Like, 'Here's a procedure that you can follow, here's some templates that you fill out.'*" P14 highlighted that these example threat models should "*show how to apply [TM] without it feeling like just explicitly writing things that you already have in your head*". These examples are significant for OSS developers who lack access to security expertise.

OSS developers need tools to automate documenting implicit project information. While many established structured TM processes can be modified to fit the needs of OSS developers, this is not true for the flexible distributed coordination necessary in OSS. For example, when describing how to simplify STRIDE, Shostack argued the most time-consuming portion of adding TM is writing out all the implicit ("ambient") information shared among project members [81]. He argued a team can skip documenting this information when first adopting TM. Unfortunately, this does not work for OSS. The decentralized nature of OSS means it is hard to get everyone in the room to do TM together, so all the implicit information cannot be shared in this way. Therefore, tools should be developed to assist developers in documenting their projects' implicit information quickly. While we can encourage developers to document more of this information, automated tools are a better way to lift the burden from OSS contributors.

Also, these tools should support sharing assumptions that affect which threats are elicited. It is unclear from prior work how tools may support assumption sharing [79], though assumptions are implicit information external contributors need.

Customized threat checklists may be a usable product of a structured TM process. A criticism of structured TM is that it requires significant effort, resulting in documentation useful only to the security team. P1 lamented, "*It's a huge discouraging thing to have to maintain this huge body of text that isn't somehow technically enforceable.*" One possible solution is the hybrid approach adopted by P3 where a structured TM process created a living document updated regularly by a small number of security experts, then used to develop a checklist for ad hoc threat identification and mitigation by developers (see Section 5.2). This offers benefits of structured processes, i.e., thoroughness, and ad hoc practices, i.e., flexibility and low overhead, while limiting burdens of documentation and expertise to the security team. This hybrid process is likely not as thorough as a fully structured TM process, but may produce sufficient security and is more likely to actually be used. Additionally, as P3 mentioned, it simplifies onboarding new developers and makes it easier to enforce mitigations as contributors must review the checklist at commit time.

8.2 Recommendations for Researchers

While we expect structured TM processes and tools provide better security, as has been the community's general assumption [102] and demonstrated by the focus on structured TM (e.g., [37, 42, 85]), this has not been systematically studied. Conversely, about half of our participants believed their ad hoc practices provided sufficient security.

Researchers should investigate ad hoc practices' outcomes. We performed a simple analysis of ad hoc practice effectiveness using OpenSSF Scorecard [2] to count vulnerabilities in the 32 Github-hosted participant projects. We found no clear relationship between vulnerabilities and the TM practices. The three projects with more-structured processes had 0, 95, and 2 vulnerabilities each. Most projects with ad hoc practices had 0-1 vulnerabilities, though some had over 30. These ad hoc practices may achieve security commensurate with or sufficiently similar to structured TM processes. However, this lack of relationship is possibly due to the diversity of other project characteristics (e.g., function, size, use cases). Our small scale qualitative analysis cannot tease apart these confounding factors to assess the outcomes of ad hoc practices. Future work is needed to investigate whether ad hoc TM is sufficient and could reveal how we can systematize practices without adding overhead or sacrificing security.

8.3 OSS Community Recommendations

Finally, we highlight one additional insight that suggested a potentially high-impact strategy for improving OSS security.

Prioritize TM adoption among projects with many forks. Among our participants, we observed some slowness in adopting new TM processes. This is expected when working with mainly volunteer contributors and decentralized organizations, with the primary focus being functionality, not security. Thus, participants were likely to continue using ad hoc practices. However, a few participants had forked projects using TM processes, meaning initial TM steps were already completed (see Section 6.5). This suggests potential in supporting highly forked projects to adopt TM and share their threat models. This could create ripple effects across the OSS ecosystem, making adoption easier, spreading TM awareness, and normalizing sharing threat models within the community.

9 Conclusion

We conducted semi-structured interviews with 25 OSS developers to understand their TM practices and challenges, how those challenges shape adoption decisions, and how to support their TM practices. Almost all participants used ad hoc TM practices. Our participants primarily developed their own ad hoc practices emphasizing flexibility, ease of use, and limited overhead. Volunteer contributors lack access to security experts, motivation, and time for TM and find it difficult to reach a consensus due to OSS's decentralized structure. Many structured TM processes can be more lightweight, such as using the STRIDE mnemonic to support as-needed review. However, improved tool support that minimizes developer effort, integrates with existing technology, and supports asynchronous work, along with example threat models reducing the need for on-team security experts, are missing. Through these recommendations, we hope to encourage greater TM use in OSS to secure the core of the software supply chain.

Acknowledgments

We thank all the interviewees for participating and supporting our research. We thank the anonymous reviewers who provided helpful comments on drafts of this paper. This project was partially supported by gifts from Cisco and MedCrypt. This research was also partially funded by VolkswagenStiftung Niedersächsisches Vorab (ZN3695). The views presented in the paper are those of the authors and do not necessarily reflect the views of any funding agencies.

Ethics Considerations

Before the four informal conversations with OSS professionals, we approached the Tufts University Social, Behavioral, and Educational Research Institutional Review Board (Tufts SBER IRB), who deemed these conversations non-human subjects research as we did not ask about participants' specific practices or report particular findings from these conversations. Considering we did not compensate these four participants for these early discussions, we avoided asking anything that could not be ascertained via public data, and we do not report on specific comments. Instead, we used their responses to shape our interview design.

Our interview study was reviewed and approved by Tufts SBER IRB. We obtained mandatory informed consent from all participants before they could proceed with the survey. At the interview start, we reminded the participant about key information in the previously signed consent form, including that we would remove from the interview transcript any mentions of their name, their project's name, or any other information that could de-anonymize associated people or projects. We asked participants to reaffirm their consent before recording the interview. They could skip any question or stop the interview anytime if they felt uncomfortable. This study was conducted in a GDPR-compliant manner and is consistent with the standards defined in the Menlo Report [31]. As mentioned in Section 3.5, though we utilized several lists of projects hosted on GitHub, we did not extract participants' emails from GitHub since using email addresses found on GitHub violates the website's acceptable use policies [7]. Instead, we searched for personal or project-related websites that contained contact information of the contributors.

The only potential harm we envision from publishing this work is that malicious hackers may target OSS projects knowing they do not perform structured TM processes that reduce vulnerabilities. However, we believe this risk is low, as there already exists a general assumption that OSS projects rarely use suggested TM practices [26, 34]. Conversely, we believe our work offers a significant benefit for OSS developers as we suggest improvements for OSS development to encourage greater adoption of TM processes. Our analysis of TM practices and challenges to TM throughout the software lifecycle allows us to make actionable recommendations for the OSS community, tool developers, companies relying on OSS, and researchers to continue securing OSS. Therefore, we believe the limited potential harms to the OSS community are outweighed by the benefits of this work to the same community.

Open Science

To enhance and support transparency, replication, meta-research, and in compliance with the open science policy, we provide the following research artifacts in online supplemental materials [3] in addition to the appendix: (1) interview guide, (2) survey questions (3) Upwork screening questions, and (4) codebook.

We do not include raw interview data, i.e. interview audio and transcripts in our replication package in compliance with data protection requirements and ethical research practices. This emphasizes our commitment to protecting participant privacy and ensuring their right to data protection. By doing this, we mitigate the risk of leaking any information that could be used to identify our participants or their projects, maybe through contextual and/or meta information. Instead, we use thematic analysis along with anonymized interview quotes to share our research findings.

References

[1] crates.io: Rust package registry. https://crates.io/.

[2] Openssf scorecard. https://scorecard.dev/.

[3] OSS threat modeling interviews supplemental materials. https://osf.io/kn5pb/?view_only=d82c73d6ca5749c38fad348806703642.

[4] OWASP Secure Coding Practices - Quick Reference Guide | Secure Coding Practices | OWASP Foundation. https://owasp.org/www-project-secure-coding-practices-quick-reference-guide/stable-en/02-checklist/05-checklist.html.

[5] What is architecture diagramming? - software & system architecture diagramming explained - aws. https://aws.amazon.com/what-is/architecture-diagramming/#:~:text=Architecture%20diagrams%20identify%20potential%20system,that%20significant%20issues%20appear%20later.

[6] Nvd - CVE-2021-44228, 2021. nvd.nist.gov/vuln/detail/CVE-2021-44228.

[7] Github acceptable use policies, 2023. https://docs.github.com/en/site-policy/acceptable-use-policies/github-acceptable-use-policies.

[8] Upwork, 2024. https://www.upwork.com/.

[9] Yasemin Acar, Michael Backes, Sascha Fahl, Simson Garfinkel, Doowon Kim, Michelle L Mazurek, and Christian Stransky. Comparing the usability of cryptographic apis. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017.

[10] Robert E. Adamson. Functional fixedness as related to problem solving: a repetition of three experiments. *Journal of experimental psychology*, 44(4):288, 1952.

[11] Edward G Amoroso. *Fundamentals of computer security technology*. Prentice-Hall, Inc., 1994.

[12] Andrew van der Stock, Brian Glas, Neil Smithline, and Torsten Gigler. OWASP Top Ten. https://owasp.org/www-project-top-ten/.

[13] Hala Assal and Sonia Chiasson. Security in the software development lifecycle. In *Proceedings of the Fourteenth USENIX Conference on Usable Privacy and Security*, 2018.

[14] Hala Assal and Sonia Chiasson. 'Think secure from the beginning': A survey with software developers. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA, 2019. Association for Computing Machinery.

[15] Moritz Beller, Radjino Bholanath, Shane McIntosh, and Andy Zaidman. Analyzing the state of static analysis: A large-scale evaluation in open source software. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, 2016.

[16] Karin Bernsmed, Daniela Soares Cruzes, Martin Gilje Jaatun, and Monica Iovan. Adopting threat modelling in agile software development projects. *Journal of Systems and Software*, 183, 2022.

[17] Hugh Beyer and Karen Holtzblatt. *Contextual Design: Defining Customer-Centered Systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1997.

[18] Eric Bodden, Sam Weber, and Laurie Williams. Empirical Evaluation of Secure Development Processes (Dagstuhl Seminar 23181). *Dagstuhl Reports*, 13(5):1–21, 2023.

[19] Grady Booch. *The unified modeling language user guide*. Pearson Education India, 2005.

[20] Rosalie Chan. Open-source developers are burning out, quitting, and even sabotaging their own projects - and it's putting the entire internet at risk, May 2022.

[21] CISA. CISA Issues Emergency Directive Requiring Federal Agencies to Mitigate Apache Log4J Vulnerabilities, December 2021. https://www.cisa.gov/news-events/news/cisa-issues-emergency-directive-requiring-federal-agencies-mitigate-apache-log4j.

[22] CISA, NSA, FBI, ACSC, NCSC-UK, CCCS, BSI, NCSC-NL, CERT NZ, and NCSC-NZ. Shifting the Balance of Cybersecurity Risk: Principles and Approaches for Security-by-Design and -Default. Technical report, April 2023.

[23] Cisco. What Is Threat Modeling?, September 2020. https://www.cisco.com/c/en/us/products/security/what-is-threat-modeling.html.

[24] Victoria Clarke and Virginia Braun. Thematic analysis. *The journal of positive psychology*, 12(3):297–298, 2017.

[25] CMS Threat Modeling Team. CMS Threat Modeling Handbook, February 2024. https://security.cms.gov/policy-guidance/threat-modeling-handbook.

[26] Dan Conn. What I found when modelling threats in the open (source), February 2023. www.youtube.com/watch?v=S1UXqPQs2Sw.

[27] Juliet Corbin and Anselm Strauss. *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage publications, 2014.

[28] Melissa Dark and Jelena Mirkovic. Evaluation Theory and Practice Applied to Cybersecurity Education. *IEEE Security & Privacy*, 13(2):75–80, March 2015.

[29] Darran Boyd. How to approach threat modeling, January 2021. https://aws.amazon.com/blogs/security/how-to-approach-threat-modeling/.

[30] Tamara Denning, Adam Lerner, Adam Shostack, and Tadayoshi Kohno. Control-alt-hack: the design and evaluation of a card game for computer security awareness and education. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013.

[31] David Dittrich and Erin Kenneally. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. Technical report, U.S. Department of Homeland Security, Aug 2012.

[32] Karl Duncker and Lynne S. Lees. On problem-solving. *Psychological monographs*, 58(5):i, 1945.

[33] Pardis Emami-Naeini, Henry Dixon, Yuvraj Agarwal, and Lorrie Faith Cranor. Exploring how privacy and security factor into iot device purchase behavior. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 2019.

[34] Erin Farr. 12 ways to improve your open source security, April 2022. developer.ibm.com/articles/12-ways-to-improve-your-open-source-security/.

[35] Danilo Favato, Daniel Ishitani, Johnatan Oliveira, and Eduardo Figueiredo. Linus's law: More eyes fewer flaws in open source projects. In *Proceedings of the XVIII Brazilian Symposium on Software Quality*, SBQS '19, 2019.

[36] Joint Task Force. Security and Privacy Controls for Information Systems and Organizations. Technical Report NIST Special Publication (SP) 800-53 Rev. 5, National Institute of Standards and Technology, December 2020.

[37] Forrest Shull. Evaluation of Threat Modeling Methodologies. In *SEI 2016 Research Review*, October 2016.

[38] Frank Nagle, Jessica Wilkerson, James Dana, and Jennifer L. Hoffman. Vulnerabilities in the Core Preliminary Report and Census II of Open Source Software. Technical report, The Linux Foundation & The Laboratory for Innovation Science at Harvard, February 2020.

[39] Sylvain Frey, Awais Rashid, Pauline Anthonysamy, Maria Pinto-Albuquerque, and Syad Asad Naqvi. The good, the bad and the ugly: A study of security decisions in a cyber-physical systems game. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018.

[40] FTC. FTC warns companies to remediate Log4j security vulnerability, January 2022. https://www.ftc.gov/policy/advocacy-research/tech-at-ftc/2022/01/ftc-warns-companies-remediate-log4j-security-vulnerability.

[41] Kelsey R. Fulton, Anna Chan, Daniel Votipka, Michael Hicks, and Michelle L. Mazurek. Benefits and Drawbacks of Adopting a Secure Programming Language: Rust as a Case Study. In *Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021)*, 2021.

[42] Kelsey R. Fulton, Daniel Votipka, Desiree Abrokwa, Michelle L. Mazurek, Michael Hicks, and James Parker. Understanding the how and the why: Exploring secure development practices through a course competition. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022.

[43] Github. Open source organizations, 2023. https://github.com/collections/open-source-organizations.

[44] Antonios Gkortzis, Dimitris Mitropoulos, and Diomidis Spinellis. VulinOSS: a dataset of security vulnerabilities in open-source systems. In *Proceedings of the 15th International Conference on Mining Software Repositories*, 2018.

[45] Leo A. Goodman. Snowball sampling. *The annals of mathematical statistics*, pages 148–170, 1961.

[46] Hana Habib, Sarah Pearman, Jiamin Wang, Yixin Zou, Alessandro Acquisti, Lorrie Faith Cranor, Norman Sadeh, and Florian Schaub. "It's a scavenger hunt": Usability of websites' opt-out and data deletion choices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020.

[47] Seth T. Hamman, Kenneth M. Hopkinson, Ruth L. Markham, Andrew M. Chaplik, and Gabrielle E. Metzler. Teaching Game Theory to Improve Adversarial Thinking in Cybersecurity Students. *IEEE Transactions on Education*, 60(3):205–211, August 2017.

[48] Julie M. Haney, Mary Theofanos, Yasemin Acar, and Sandra Spickard Prettyman. "We make it a big deal in the company": Security mindsets in organizations that develop cryptographic products. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, 2018.

[49] Michael Howard and Steve Lipner. *The security development lifecycle*, volume 8. Microsoft Press Redmond, 2006.

[50] Nasif Imtiaz, Brendan Murphy, and Laurie Williams. How do developers act on static analysis alerts? An empirical study of coverity usage. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019.

[51] James Ritchey. Threat Modeling, May 2024. https://handbook.gitlab.com/handbook/security/product-security/application-security/threat-modeling/.

[52] Brittany Johnson, Yoonki Song, Emerson Murphy-Hill, and Robert Bowdidge. Why Don't Software Developers Use Static Analysis Tools to Find Bugs? In *2013 35th International Conference on Software Engineering (ICSE)*, 2013.

[53] Gary A Klein. *Sources of power: How people make decisions*. MIT press, 2017.

[54] Sabrina Klivan, Sandra Höltervennhoff, Rebecca Panskus, Karoly Marky, and Sascha Fahl. Everyone for themselves? A qualitative study about individual security setups of open source software contributors. In *45th IEEE Symposium on Security and Privacy (SP)*, 2024.

[55] Loren Kohnfelder and Praerit Garg. The threats to our products, April 1999. shostack.org/files/microsoft/The-Threats-To-Our-Products.docx.

[56] Klaus Krippendorff. Reliability in content analysis: Some common misconceptions and recommendations. *Human communication research*, 30(3):411–433, 2004.

[57] Raula Gaikovina Kula, Daniel M. German, Ali Ouni, Takashi Ishio, and Katsuro Inoue. Do developers update their library dependencies? *Empirical Software Engineering*, 23(1):384–417, February 2018.

[58] Qing Li and Yu-Liu Chen. Data flow diagram. In *Modeling and Analysis of Enterprise and Information Systems*, pages 85–97. Springer, 2009.

[59] Microsoft. Network diagram and mapping software | microsoft visio. https://www.microsoft.com/en-us/microsoft-365/visio/network-diagrams.

[60] Jaron Mink, Harjot Kaur, Juliane Schmüser, Sascha Fahl, and Yasemin 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 *32nd USENIX Security Symposium (USENIX Security 23)*, August 2023.

[61] Mozilla. Rapid Risk Assessment (RRA), October 2019. https://infosec.mozilla.org/guidelines/risk/rapid_risk_assessment.html.

[62] Alena Naiakshina, Anastasia Danilova, Eva Gerlitz, and Matthew Smith. On conducting security developer studies with cs students: Examining a password-storage study with cs students, freelancers, and company developers. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020.

[63] A Norman Donald. *The design of everyday things*. MIT Press, 2013.

[64] Daniela Seabra Oliveira, Tian Lin, Muhammad Sajidur Rahman, Rad Akefirad, Donovan Ellis, Eliany Perez, Rahul Bobhate, Lois A. DeLong, Justin Cappos, and Yuriy Brun. API blindspots: Why experienced developers write vulnerable code. In *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, August 2018.

[65] Opensource.com. Open source organizations, 2023. https://opensource.com/resources/organizations.

[66] Opensource.com. The open source way, 2023. https://opensource.com/open-source-way.

[67] OWASP. Threat Modeling, April 2024. https://cheatsheetseries.owasp.org/cheatsheets/Threat_Modeling_Cheat_Sheet.html.

[68] Hernan Palombo, Armin Ziaie Tabari, Daniel Lende, Jay Ligatti, and Xinming Ou. An ethnographic understanding of software (in) security and a co-creation model to improve secure software development. In *Proceedings of the Sixteenth USENIX Conference on Usable Privacy and Security (SOUPS 2020)*, USA, 2020.

[69] Serena Elisa Ponta, Henrik Plate, and Antonino Sabetta. Detection, assessment and mitigation of vulnerabilities in open source dependencies. *Empirical Software Engineering*, 25(5):3175–3215, September 2020.

[70] Emilee Rader, Samantha Hautea, and Anjali Munasinghe. "I have a narrow thought process": Constraints on explanations connecting inferences and Self-Perceptions. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*, August 2020.

[71] Maria Riaz, Jason King, John Slankas, Laurie Williams, Fabio Massacci, Christian Quesada-López, and Marcelo Jenkins. Identifying the implied: Findings from three differentiated replications on the use of security requirements templates. *Empirical Software Engineering*, 22(4):2127–2178, August 2017.

[72] William B. Rouse and Nancy M. Morris. On looking into the black box: Prospects and limits in the search for mental models. *Psychological Bulletin*, 100(3):349–363, 1986.

[73] Andrew Ruef, Michael Hicks, James Parker, Dave Levin, Michelle L Mazurek, and Piotr Mardziel. Build it, break it, fix it: Contesting secure development. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 690–703, 2016.

[74] Benjamin Saunders, Julius Sim, Tom Kingstone, Shula Baker, Jackie Waterfield, Bernadette Bartlam, Heather Burroughs, and Clare Jinks. Saturation in qualitative research: exploring its conceptualization and operationalization. *Quality & quantity*, 52:1893–1907, 2018.

[75] Bruce Schneier. Attack trees. *Dr. Dobb's journal*, 24(12):21–29, 1999.

[76] Brook S.E. Schoenfield. *Secrets of a Cyber Security Architect*. Auerbach Publications, 2019.

[77] Muhammad Shahzad, Muhammad Zubair Shafiq, and Alex X. Liu. A large scale exploratory analysis of software vulnerability life cycles. In *2012 34th International Conference on Software Engineering (ICSE)*, pages 771–781, 2012.

[78] Nataliya Shevchenko, Brent R. Frye, and Carol Woody. Threat Modeling for Cyber-Physical System-of-Systems: Methods Evaluation. Technical report, Carnegie Mellon University Software Engineering Institute, September 2018. Section: Technical Reports.

[79] Zhenpeng Shi, Kalman Graffi, David Starobinski, and Nikolay Matyunin. Threat modeling tools: A taxonomy. *IEEE Security & Privacy*, 20(4):29–39, 2022.

[80] Adam Shostack. *Threat modeling: Designing for security*. 2014.

[81] Adam Shostack. Fast, Cheap and Good: An Unusual Trade-off Available in Threat Modeling, December 2021.

[82] B. Shreeve, J. Hallett, M. Edwards, K. M. Ramokapane, R. Atkins, and A. Rashid. The best laid plans or lack thereof: Security decision-making of different stakeholder groups. *IEEE Transactions on Software Engineering*, 48(05):1515–1528, 2022.

[83] Benjamin Shreeve, Joseph Hallett, Matthew Edwards, Pauline Anthonysamy, Sylvain Frey, and Awais Rashid. "So If Mr Blue Head Here Clicks the Link..." Risk Thinking in Cyber Security Decision Making. *ACM Trans. Priv. Secur.*, 24(1), 2020.

[84] Sonatype. 8th Annual State of the Software Supply Chain. Technical report, 2023. https://www.sonatype.com/state-of-the-software-supply-chain/introduction.

[85] Rock Stevens, Daniel Votipka, Elissa M. Redmiles, Colin Ahern, Patrick Sweeney, and Michelle L. Mazurek. The battle for new york: A case study of applied digital threat modeling at the enterprise level. In *27th USENIX Security Symposium*, August 2018.

[86] Synopsys. Analyst report open source security and analysis report. Technical report, 2024. www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html.

[87] Tatum Hunter and Gerrit De Vynck. The 'most serious' security breach ever is unfolding right now. Here's what you need to know., December 2021. https://www.washingtonpost.com/technology/2021/12/20/log4j-hack-vulnerability-java/.

[88] Microsoft Software Development Lifecycle Team. The stride per element chart. https://www.microsoft.com/en-us/security/blog/2007/10/29/the-stride-per-element-chart/, 2007.

[89] Ronald Thompson, Madeline McLaughlin, Carson Powers, and Daniel Votipka. "there are rabbit holes I want to go down that I'm not allowed to go down": An investigation of security expert threat modeling practices for medical devices. In *33rd USENIX Security Symposium*, August 2024.

[90] Tony UcedaVélez and Marco M. Morana. *Risk Centric Threat Modeling: process for attack simulation and threat analysis*. John Wiley & Sons, 2015.

[91] Warda Usman, Jackie Hu, McKynlee Wilson, and Daniel Zappala. Distrust of big tech and a desire for privacy: Understanding the motivations of people who have voluntarily adopted secure email. In *Nineteenth Symposium on Usable Privacy and Security (SOUPS 2023)*, 2023.

[92] Dimitri Van Landuyt and Wouter Joosen. A descriptive study of assumptions in stride security threat modeling. *Software and Systems Modeling*, pages 1–18, 2021.

[93] Stef Verreydt, Koen Yskout, Laurens Sion, and Wouter Joosen. Threat modeling state of practice in dutch organizations. In *Twentieth Symposium on Usable Privacy and Security (SOUPS 2024)*. USENIX Association, August 2024.

[94] Daniel Votipka, Desiree Abrokwa, and Michelle L. Mazurek. Building and validating a scale for secure software development self-efficacy. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020.

[95] Daniel Votipka, Kelsey R. Fulton, James Parker, Matthew Hou, Michelle L. Mazurek, and Michael Hicks. Understanding security mistakes developers make: Qualitative analysis from build it, break it, fix it. In *29th USENIX Security Symposium*, 2020.

[96] Dominik Wermke, Noah Wöhler, Jan H Klemmer, Marcel Fourné, Yasemin Acar, and Sascha Fahl. Committed to trust: A qualitative study on security & trust in open source software projects. In *2022 IEEE Symposium on Security and Privacy (SP)*, 2022.

[97] David A. Wheeler. Secure software development fundamentals, 2023.

[98] Chamila Wijayarathna and Nalin AG Arachchilage. Why Johnny Can't Store Passwords Securely? A Usability Evaluation of Bouncycastle Password Hashing. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering*, 2018.

[99] Wenjun Xiong and Robert Lagerström. Threat modeling – a systematic literature review. *Computers & Security*, 84:53–69, 2019.

[100] Jie Zhang, Haoyu Bu, Hui Wen, Yu Chen, Lun Li, and Hongsong Zhu. When llms meet cybersecurity: A systematic literature review, 2024.

[101] Shikun Zhang, Yuanyuan Feng, Yaxing Yao, Lorrie Faith Cranor, and Norman Sadeh. How usable are ios app privacy labels? *Proceedings on Privacy Enhancing Technologies*, 2022.

[102] Zoe Braiterman, Adam Shostack, Jonathan Marcil, Stephen de Vries, Irene Michlin, Kim Wuyts, Robert Hurlbut, Brook S.E. Schoenfield, Fraser Scott, Matthew Coles, Chris Romeo, Alyssa Miller, Izar Tarandach, Avi Douglen, and Marc French. Threat Modeling Manifesto.

A Interview Questions

Section 1: Background and Introduction.

1. [**Experience**] In your survey, you mentioned you have been involved in OSS for [*years from survey*] years. Can you tell us a little more about your experience?

2. [**Project details**] Can you tell us a bit about your project(s)? For e.g. these could be security-related projects you most contributed to / your most recent project.

- **2.1.** What is it about?
- **2.2.** What is your role in this project? (i.e. What are your responsibilities and duties?)
- **2.3.** Is your project completely or partly open source?
- **2.4.** [If the project is supported by a company] Roughly what percentage of core contributors are within [*company*] compared to external contributors?
  - **2.4.1.** What does that look like overall for the project, i.e. both core and non-core contributors?

3. [**Project security relevance**] Do you consider security an important aspect for you in general and for your project?

- **3.1.** [*If yes for 3.*] Why do you care about security?
- **3.2.** [*If no for 3.*] Why not?

Section 2: Process Followed.

4. [**Scope of threat model - Security Objectives**] What are the "security objectives" for your projects? (i.e., what is the level of security or security properties you want to achieve?)

5. [**what am I building**] How do you visualize the system you are building? For e.g., this could include everything from diagramming to making a mental model etc.
(*If prompt needed*): How do you visualize your system to keep track of where you have looked for threats? Which elements of your system do you look at to find threats?

6. [**What can go wrong? - Threat Identification in {*Design/Implementation*}** *] Now we are going to ask a few questions about identifying threats when you are {*designing / writing*} the software, or while {*designing / writing*} a specific component.
When {*designing software / writing code*}, how do you look for threats/vulnerabilities/security issues? Can you give us an example?

- **6.1.** [**Documentation**] {*During software design / While writing code*}, do you keep a record of the identified threats using well-defined documentation for future reference?
- **6.2.** [**Communication**] {*During software design / While writing code*}, how do you communicate the identified threats to other people on the project?
  - **6.2.1.** [***if software library***] How do you communicate the identified threats to developers who use the library?
  - **6.2.2.** [***if external contributors***] How do you communicate the identified threats to external contributors?
    - **a.** Do you have a policy for what gets communicated to external contributors?
- **6.3.** [**Responsibility**] Whose responsibility is it to think about/ identify threats {*during software design / while writing code*}?
- **6.4.** [**Resources**] Do you rely on any resources (e.g. books, websites, training, blogs, specific people etc.) to help you with threat identification {*during software design / while writing code*}?
- **6.5.** [**Challenges**] What are the most challenging aspects of your threat identification process {*during software design / while writing code*}?
(*If prompt needed*): Are there any resources you lack? Better software visualization? More hours? More structure?

*We asked the question block "what can go wrong" both for design and implementation stages. We first asked the entire question block probing for practices during design. We then ask the same set of questions probing for practices during implementation. Changes in wording are marked as {*during software design / while writing code*} or similar text.

6.5.1. [*if project supported by company & external contributors involved*] You mentioned that your project has a mix of internal contributors from your company and external contributors. Do you find any challenges to threat identification {*during software design / while writing code*} due to the in-group and out-group structure?

7. [**What am I going to do about it? - Threat Mitigation in {*Design / Implementation*}**[*]] When {*designing software / writing code*}, how do you decide what to do about security issues you identified? Can you give an example?

7.1. In what order do you think about mitigations? How do you triage these threats? (*If prompt needed*): i.e., do you think of an issue and then a solution one-by-one, or all the issues first and then what to do about each of them, etc.?)

7.1.1. Do you categorize the identified security issues in some way? [*If yes*] Why do you categorize them?

7.2. [**Documentation**] Do you keep a record of these solutions to security issues using well-defined documentation for future reference?

7.3. [**Communication**] How do you communicate these solutions to security issues to other people on the project?

7.3.1. [*If software library*] How do you communicate the solutions to security issues to developers who use the library?

7.3.2. [*If external contributors*] How do you communicate these mitigation strategies to external collaborators to ensure their commits are consistent with this approach?

7.4. [**Responsibility**] Who decides what is a valid issue that should be fixed or which fixes will be used?

7.4.1. Whose responsibility is to come up with solutions/fixes?

7.5. [**Resources**] Do you rely on any resources (e.g. books, websites, training, blogs, specific people etc.) to help you with threat mitigation?

7.6. [**Challenges**] What are the most challenging aspects of the mitigation process?

7.6.1. [*if project supported by company & external contributors involved*] You mentioned that your project has a mix of internal contributors from your company and external contributors. Do you find any challenges to the threat mitigation process {*during software design / while writing code*} due to the in-group and out-group structure?

8. **[Threat identification & mitigation - other stages]** Do you identify or mitigate threats at any other stages?

9. **[What can go wrong? - Frequency/Revising threat identification & mitigation]** Now we are going to ask about when you make changes to your code. By "changes," we mean: changes to functionality, changes in involved parties, changes due to a vulnerability being found etc.. Do your threat identification and mitigation processes resume when making a change?

9.1. [*If yes for 9.*] Could you elaborate?

9.2. Does this happen with every change or just for specific changes?

10. **[Did I do a good enough job? - Validation]** Is there any process for determining you've met your security objectives? (Can you give us an example?)

10.1. Is a particular person responsible for determining that you've met your security objectives?

Section 3: Threat Modeling.

- **[Introducing Threat Modeling]** The questions we have been asking are about your process of finding threats and mitigations. These processes are frequently called "Threat Modeling".

[*]We follow the same strategy we used previously for the "what can go wrong" block for the current questions block "What am I going to do about it?"

11. Have you heard of Threat Modeling?

11.1. [*If yes for 11.*] How would you define threat modeling?

11.2. Do you feel the processes you described above meet that definition of "Threat Modeling?"

[Our definition] We define Threat Modeling as the general process of finding threats to a system, and deciding what to do about those threats. We don't believe it must be a rigorous, structured process as some people define it, so we have so far avoided using that term.

12. [*if they do not mention any formal TM techniques above*] Do you implement any kind of formal threat modeling in your project? (By "formal," we mean structured approach like STRIDE, PASTA, LINDDUN, etc.)

12.1. [*If no for 12.*] Why do you not use a formal threat modeling process?

12.2. [*If no for 12.*] Have you considered using it in the past?

Section 4: Adoption.

13. [*If they do implement TM in any way*] [**Reasons**] What are your reasons for using threat modeling (or a threat modeling-like process you just mentioned)?

14. [*If they do implement TM in any way*] [**Adoption**] Why did you/your team decide to adopt threat modeling (or a threat modeling-like process you just mentioned)?

14.1. Were you a part of this group when this process was adopted?

15. [**Adoption**] Was it hard to convince the necessary people in your group to adopt this threat modeling-like process?

15.1. What concerns did they have?

15.2. What were they excited about?

16. [**Adoption**] What would you tell someone in your position in a different OSS project that is also thinking about adopting a threat modeling process?

Section 5: Challenges.

17. [*If they do implement TM only to some extent but not completely*] Based on your previous responses, we see that you implement some threat modeling in your projects - Do you think/believe that the extent to which you implement threat modeling in your project is sufficient to ensure security?

17.1. [*if no* for 17.] Do you find any challenges or barriers that prevent you from completely implementing threat modeling in your project?

18. [*if project supported by company & external contributors involved*] Do you encounter any other challenges with implementing threat modeling (or a threat-modeling like process) due to having internal and external contributors?

19. How easy is it for you to find solutions to any problems you encounter while implementing threat modeling (or a threat modeling-like process you just mentioned)?

20. [*If no TM*] Do you find any challenges or obstacles that prevent you from utilizing any threat modeling in your project?

Section 6: Improvements.

21. [*Ask only if they have a familiarity with TM*] In your opinion, how do you think OSS projects can be better supported in implementing threat modeling?

21.1. What additional resources would be helpful for implementing threat modeling in your open source project(s)?

21.1.1. [*if they do informal TM]* If you were to implement formal TM in your project, what resources would you find helpful?

22. What would you do differently if you were to implement threat modeling in another open source project?

Section 7: End of Interview.

23. Before we finish, is there anything that we did not discuss that you think would be helpful for us to know?