Who's Pushing the Code? An Exploration of GitHub Impersonation

Yueke Zhang[1], Anda Liang[1], Xiaohan Wang[1], Pamela Wisniewski[1], Fengwei Zhang[2], Kevin Leach[1], Yu Huang[1]
[1]*Vanderbilt University, Nashville, USA*
{yueke.zhang, anda.liang, xiaohan.wang.1, pamela.wisniewski, kevin.leach, yu.huang}@vanderbilt.edu
[2]*Southern University of Science and Technology, Shenzhen, China*
zhangfw@sustech.edu.cn

***Abstract*—GitHub is one of the largest open-source software (OSS) communities for software development and collaboration. Impersonation in the OSS communities refers to the malicious act of assuming another user's identity, often aiming to gain unauthorized access to code, manipulate project outcomes, or spread misinformation. With several recent real-world attacks resulting from impersonation, this issue is becoming more and more concerning within the OSS community. We present the first exploration of the impact of impersonation in GitHub. Specifically, we conduct structured interviews with 17 real-world OSS contributors about their perception of impersonation and corresponding mitigations.**

Our study reveals that, in general, GitHub users lack awareness of impersonation and underestimate the severity of its implications. After witnessing a demo of impersonation, they show significant concern for the OSS community. Meanwhile, we also demonstrate that the current best practices (i.e., commit signing) that might mitigate impersonation must be improved to encourage use and adoption. We also present and discuss participant perceptions of potential ways to mitigate GitHub impersonation.

We collect a dataset comprising 12.5 million commits to investigate the current status of impersonation. Interestingly, we find out that currently impersonation cannot be easily detected. We observe that existing commit histories treat impersonation behavior identically to pull request events, resulting in a lack of detection methods for impersonation.

***Index Terms*—Open-Source Software, Impersonation**

I. Introduction

Over the past years, GitHub has evolved to become more than just a platform for open-source software (OSS) development but also a social ecosystem that democratizes the field, enabling contributions from diverse backgrounds and skill levels [1]. In particular, GitHub has a dramatic impact on software development, education, and social networking among developers [2], [3]. It is also a social platform for OSS developers to communicate and share knowledge, as well as develop their social profiles [4]–[6].

GitHub adopts a flexible approach to commits as a natural consequence of emphasizing collaboration. Since it is based on git, users can freely change their commit email, allowing developers to commit not limited to a single account [7]. However, this flexibility also presents challenges related to identity issues within the community. With just one git command, a developer can assume anyone's identity to make commits. This commit mechanism can lead to unexpected malicious behaviors. As illustrated in Figure 1, an attacker ("Eve") can configure her email to pretend to be a credible developer, enabling them to introduce malicious code into projects. We refer to such behavior as *impersonation*. It is difficult to determine who is pushing the code in a commit under impersonation.

Real-world attacks have exploited impersonation in GitHub in very recent years. A newly-emerged supply chain attack used impersonation to compromise GitHub repositories [8] in 2022. More alarmingly, a North Korean group has implemented malware by creating repositories relating to blockchain, cryptocurrency, and online gambling in 2023. These repositories involve the impersonation of reputable developers committing to them [9]. In turn, victim users are deceived into downloading the malicious software associated (in the form of malicious npm packages in the case of [9]). With the growing security and privacy threats posed by impersonation, very little work has explored warning GitHub users about the risks associated with impersonation [10].

Existing research on OSS security focuses on issues like API leakage and metadata manipulation, overlooking the impersonation issue [11]–[15]. Though features, like commit signing, could potentially mitigate impersonation, they may not be widely used nor practical enough for wide acceptance. Impersonation is a natural consequence of the git architecture, and indeed GitHub already has policies forbidding impersonation [16]. However, we still lack an understanding of how impersonation influences developers in OSS communities, especially on GitHub. We present the first evaluation of developers' perception and awareness of impersonation and explore potential mitigation strategies for this issue. We investigate (1) the awareness of impersonation among GitHub users, (2) the level of concern among developers regarding impersonation, (3) the widespread acceptance of official practices (e.g., commit signing), (4) strategies to mitigate impersonation from the perception of GitHub users, and (5) the feasibility of detecting impersonation.

In our study, we conducted semi-structured interviews with 17 GitHub contributors. We reveal that the majority of participants were unaware of impersonation and underestimated its severity until encountering examples of impersonation.

1558-1225/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSE55347.2025.00065

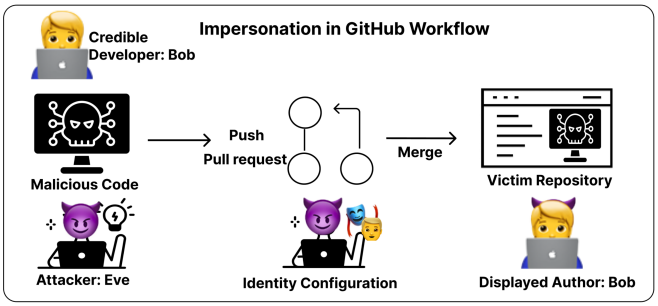

Fig. 1. The figure illustrates an impersonation scenario on GitHub. The attacker Eve aims to introduce malicious code into a repository; she can pretend to be a credible developer "Bob" by configuring her commit email to match Bob's. The tactic causes Eve's potentially malicious commits to appear as though they were created by Bob, in turn gaining trust from individuals in a victim repository.

We also collected data encompassing 12.5 million commits, from which we measured that 56% of the commits did not use commit signing. Furthermore, we investigate the reasons why developers do not adopt commit signing. Even though impersonation is a concern for participants, we also discovered that there is currently no singularly effective method to identify impersonation behavior within GitHub commit history, primarily due to the similarity that manifests between pull requests and impersonation in git commits.

In summary, our study is the first to investigate GitHub impersonation both from the perspective of developers and data accessible from GitHub. Specifically, we make the following contributions:

1) We conduct the first study to investigate impersonation on GitHub. In interviews with GitHub users, participants expressed heightened concern regarding impersonation.
2) We also assess the usage patterns of existing best practices for mitigating impersonation (i.e., commit signing).
3) We illustrate the participants' perceptions regarding the impact that impersonation can have on GitHub and explore potential mitigation strategies based on their own experiences.
4) We made the first attempt to uncover the possibility and challenge to detect the impersonation and also provide guidance for future work.

The remainder of this paper is organized as follows: Section 2 presents a motivating example, Section 3 analyzes related work, Section 4 describes our research questions, empirical methods design, and impersonation detection attempt, Section 5 presents the results, Section 6 discusses the implications, Section 7 discusses limitations, and Section 8 concludes the paper. We share the interview script and answers from participants of this study[1].

II. Motivating Example: Doppelganger Demo

We demonstrate how impersonation can be implemented in GitHub and can cause security issues through the Doppelganger demo[2]. In Figure 1, we assume an attacker, Eve, intends to introduce malicious code to a repository by impersonating the identity of a credible developer, Bob.

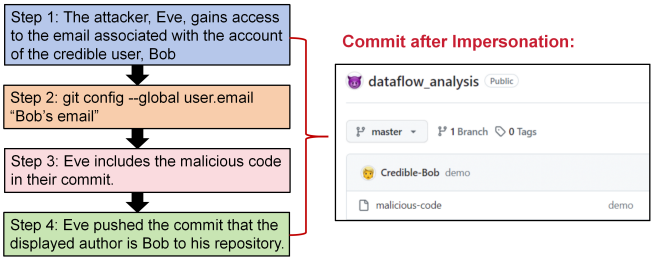

Fig. 2. A motivating example showing impersonation of a legitimate user.

In Figure 2, we illustrate the process of impersonation. Initially, Eve finds Bob's commit history in GitHub and locates the email associated with his account. Then, Eve uses the git command `git config --global user.email [Bob's email]` to set her git identity as Bob's. Eve can create a fork of the target repository and use the regular push command to push commits to her fork. Similar to the right side of Figure 2, the displayed author of the malicious code is now Bob, even though Eve is the actual contributor of the code. In this paper, we show participants this process in a structured interview.

III. Related Work

This section first discusses software vulnerabilities in open-source software (OSS) and how patches can introduce serious vulnerabilities in OSS projects. Next, we focus on existing exploitation of OSS security issues, including metadata vulnerabilities and supply chain attacks. Finally, we introduce the issue of impersonation and user identification problems on GitHub.

A. Software Vulnerability in OSS

Several existing studies focus on vulnerabilities introduced by patches in commits. According to research investigating vulnerability-introducing commits on GitHub, less experienced developers are 1.8 to 24 times more likely to contribute a vulnerable patch compared to experienced developers [17]. Additionally, over 40% of vulnerabilities are introduced by new contributors to a repository, a substantially higher rate than that of original contributors [18].

Moreover, some commits intentionally introduce malicious code. For instance, hypocrite commits appear to fix minor issues, while serious vulnerabilities are in fact introduced [19]. Presently, there are recurring attacks targeting open-source software (OSS) projects aimed at compromising the entire supply chain [20]. The objective of such attacks is to plant malicious code into open-source projects, which can then be executed within downstream projects' contexts.

[1]https://github.com/zyueek/ICSE25_github

[2]The word doppelgänger is German for "double walker;" it commonly refers to a person who is almost identical to another.

As the influence of the OSS community grows, vulnerability commits have a greater impact than ever before. Impersonation, as an internal issue within Git mechanisms, will become increasingly important in the future.

B. Exploitation in OSS

Much of society's critical infrastructure and the ability to innovate depends on the health of OSS [21]. However, security issues are prevalent in OSS, comprising approximately 3% of all issues [22]. More specifically related to our work, security-related discussions account for approximately 10% of all discussions on GitHub [23]. Exploitations of GitHub commonly fall into two main categories: *Infrastructure Vulnerabilities* and *Supply Chain Attacks*.

Infrastructure Vulnerabilities includes issues like buffer overflows, SQL injections, cross-site scripting (XSS), metadata manipulation, and remote code execution vulnerabilities launched against GitHub itself. Morton et al. [24] explored code-related risks, including buffer overflows and SQL injections. In addition, a defense scheme that mitigates metadata manipulation attacks is proposed by maintaining a cryptographically signed log of relevant developer actions [15].

Supply Chain Attacks include malicious code inserted into open-source components, which then gets propagated when other software depends on these components. For instance, Ohm et al. [25] focus on supply chain issues, especially the risks tied to malicious code insertion. In addition, Alexopoulos et al. [26] raised concerns that the openness of OSS might amplify these issues.

Despite these main categories discussed above, a recent study [27] has provided a review of threat modeling techniques tailored for OSS, which offers more insights into categorizing security issues in OSS. However, there is a lack of research exploring how human developers interact with (or are deceived within) within OSS contexts.

C. Impersonation issues in OSS

Many OSS security issues are related to identity management and cryptography [22]. Git itself also provides some identity authentication features. One important instance is *commit signing* [28].

However, with a service like GitHub, the server creates a commit object it cannot sign on behalf of the user, as it lacks the cryptographic key material needed for the signature. To improve this, *le-git-imate* [29] provides security guarantees compatible with Git's standard commit signing mechanism, which can defend against high-impact attacks on web-based Git repositories. Another mechanism, *push certificates*, introduced in version 2.2.0 of Git, allows a user to digitally sign the reference that points to a pushed object. However, push certificates are designed for out-of-band auditing. As a result, push certificates are rarely used in practice.

Despite the importance and challenge of identity authentication using Git, GitHub allows developers to pretend to use aliases for the commits. In Git, the identity information attached to each commit, including the name and email address of the user, is determined by the configurations set by the user in their local Git environment. To identify the authorship of code changes (commits) in Open-source repositories, existing work [30] has found that within around 38 million author IDs, there are around 14.8 million IDs to have an alias, which belong to 5.4 million different developers, with the median number of aliases being 2 per developer.

IV. Study Design

Our study comprises two stages to investigate the impact of impersonation on GitHub: (1) intervention interview and (2) impersonation detection attempt. Our study seeks to answer the following research questions:

- RQ1: To what extent are GitHub developers aware of and concerned about impersonation? How does their perception change before and after witnessing the Doppelganger demo?
- RQ2: How widely is the current general practice (i.e. commit signing) accepted and used among developers?
- RQ3: To what extent can we detect the potential impersonation on GitHub?
- RQ4: How can GitHub users be safeguarded against impersonation based on user perception?

A. Intervention Interview

Following best practices in the HCI literature, we conduct structured interviews with 17 participants to assess their awareness and perception of the severity and impact of impersonation on GitHub. We discuss the recruitment, participant demographics, and interview process in this subsection [31]. We refer to this as the "intervention interview" because we show participants a demonstration of GitHub impersonation and then interview them about their perceptions.

1) Data Collection and Recruitment: We use GitHub's REST API to collect user and their commit information from GitHub [32]. First, we queried the GitHub API using randomly generated two-letter strings (e.g., 'ae') to retrieve lists of repositories ranked by star count in descending order [33] and retrieve the 1,000 most starred repositories for each two-letter string [34]. From 10 batches of such random searches, we compiled the most-starred projects, resulting in 9,810 unique repositories after removing duplicates. We gathered the commit history of each repository between July 1st, 2022, and September 1st, 2022 [34][3]. Subsequently, we extract the contributor list for these repositories and retrieve all the commits made by each contributor during the specified timeframe [36]. To gather information about each contributor, we use the contributor event API, which provides insights into their behavior and activities within the repository [37].

Additionally, the repository commit API allows us to collect comprehensive data on every commit made within the repository [38]. From the contributor API, we obtain each contributor's email address [39], [40]. We reviewed the collected repositories to identify users employing multiple or

[3]GitHub's API is both rate limited [35] and precludes automatically gathering larger amounts of data about each repository.

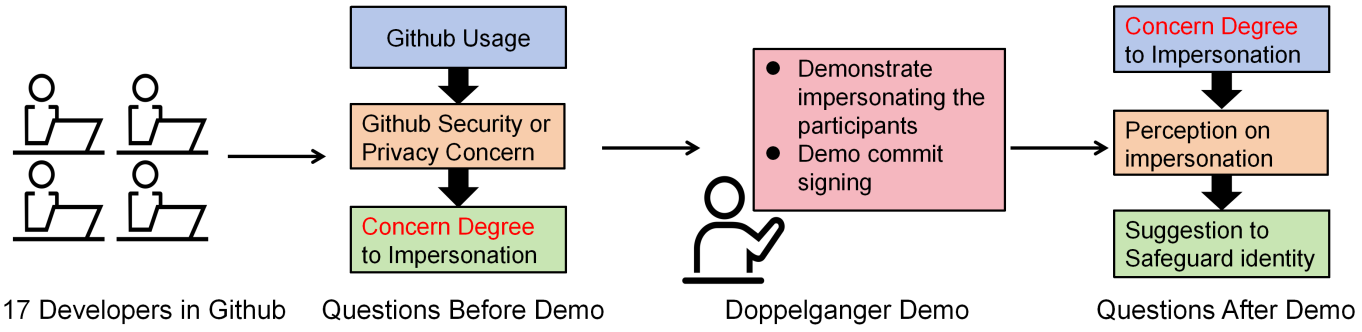

Fig. 3. A diagram showing our Interview design

unassociated emails (compared to their public GitHub user email) in unverified commits, resulting in 373 users. Then, we emailed 373 contributors and finally recruited 17 participants, with a response rate of 4.6%.

2) Participant Demographics: The demographic information is detailed in Table I, labeled P1 through P17. Participants include 12 males and 5 females. The participants' coding experience ranges from beginners with less than one year to developers with over five years of experience. The GitHub usage frequency varies significantly among participants — 6 out of the 17 participants use GitHub nearly every day. Geographically, the participants primarily contributed from the United States, China, and Canada.

Moreover, we also grouped our participants based on their experience and their background for further analysis on individual differences in this study. Specifically, we first consider participants' prior experience in open-source:

- **Novice group**: The group includes participants with less than three years of OSS experience. (P1, P2, P3, P4, P5, P6, P7, P8, P10, P11, P12, and P16)
- **Expert group**: The group includes participants with more than three years of OSS experience and use GitHub every week. (P9, P13, P14, and P15)

Then, we also consider the participants' occupations:

- **Non-industry group**: The group includes participants who are not working in the computer science industry. (P4, P7, P13, P14, P15, and P16)
- **Industry group**: The group includes participants who are working in the computer science industry. (P1, P2, P3, P5, P6, P8, P9, P10, P11, and P12)

P17 has previously worked at GitHub and is familiar with the impersonation process, and provided very insightful opinions for the study from an internal view (i.e., as a GitHub employee). Due to P17's extensive knowledge of OSS impersonation, we will not include his perspectives in the general discussion but present his opinions separately in the discussion section instead.

3) Protocol: We conducted and recorded these interviews remotely via Zoom, ranging from 25 to 40 minutes per participant. Three of the authors conducted the interviews. Before the interview, all participants signed an informed consent form and completed a demographic survey through email. Participants were required to be over 18 years of age and express comfort in communicating in English. Participants were briefed on the study's context and purpose. We offer a $20 Amazon gift card as a participant incentive[4]. The interview format follows widely used intervention-based studies [41], which we leverage to understand how participants perceive impersonation on GitHub.

4) Interview Design: As depicted in Figure 3, our interview consisted of three steps. First, we asked participants a series of questions before the demo, including their reasons for using GitHub, their concerns regarding privacy and security issues, their awareness of impersonation, and their level of concern regarding impersonation, without revealing our demo. We also use a 10-point Likert scale to gauge their degree of concern regarding impersonation [42].

Next, we presented participants with the Doppelganger Demo as an intervention. Interviewers share the screen during the demo, showing the git commands used during impersonation. If a participant is willing to engage actively in this process, we request their email address solely to demonstrate impersonation within the context of our study. Then, similar to the process outlined in Section II, participants will navigate through the entire impersonation process, and the interviewers impersonate their identity. Finally, participants observe their identity displayed in the interviewer's repository as the commit author, even though the interviewer wrote and pushed the code. If participants are unwilling to provide us with their GitHub email, we will only demonstrate the impersonation process and will not impersonate them.

After the demo, we asked participants to reassess their level of concern to determine again if there was a shift. We also inquired about the perceived influence of impersonation on GitHub and elicited suggestions from participants on how developers can safeguard themselves.

5) Qualitative and Quantitative analysis in interview data: We transcribed the interview conversations into text using third-party software and conducted inductive thematic analysis over multiple phrases [43].

1) First, we remove any irrelevant information, filler words, or repetitions that do not contribute to the analysis.

[4]This study is approved by the IRB 231741 from Vanderbilt University

TABLE I
DEMOGRAPHIC INFORMATION OF THE PARTICIPANTS.

ID	Sex	Education (Completed or Ongoing)	Coding Exp	OSS Exp	Github Usage	Occupation	Location
P1	Male	PhD or higher	3-5 years	1-3 years	Almost Every day	CS or related field student	US
P2	Male	Master's Degree	1-3 years	1-3 years	2-3 times a week	CS or related field student	US
P3	Male	Master's Degree	5 years or more	1-3 years	2-3 times a month	CS or related field student	China
P4	Male	Bachelor's Degree	1-3 years	1-3 years	2-3 times a week	Software Developer/Engineer	US
P5	Male	PhD or higher	3-5 years	1-3 years	2-3 times a week	CS or related field student	US
P6	Female	PhD or higher	3-5 years	1-3 years	2-3 times a week	CS or related field student	China
P7	Female	PhD or higher	3-5 years	1-3 years	Almost Every day	Machine Learning Engineer	China
P8	Female	Master's Degree	3-5 years	Below 1 year	2-3 times a month	CS or related field student	US
P9	Male	PhD or higher	5 years or more	5 years or more	Almost Every day	CS or related field student	US
P10	Male	PhD or higher	5 years or more	3-5 years	2-3 times a month	CS or related field student	US
P11	Female	Bachelor's Degree	3-5 years	1-3 years	2-3 times a month	CS or related field student	US
P12	Male	Master's Degree	3-5 years	1-3 years	2-3 times a month	CS or related field student	US
P13	Male	Bachelor's Degree	5 years or more	3-5 years	Almost Every day	Software Developer/Engineer	US
P14	Female	Master's Degree	5 years or more	5 years or more	2-3 times a week	Software Developer/Engineer	Canada
P15	Male	Bachelor's Degree	5 years or more	5 years or more	2-3 times a week	Software Developer/Engineer	US
P16	Male	Master's Degree	5 years or more	Below 1 year	Almost Every day	Machine Learning Engineer	US
P17*	Male	Bachelor's Degree	3-5 years	1-3 years	Almost Every day	Software Developer/Engineer	Germany

2) Next, we generate code responses to specific questions or thematic segments that emerge during the interviews.
3) Subsequently, we assign codes or labels to different segments based on their content.
4) Finally, we organized the code into meaningful themes.

The theoretical saturation was reached at the 13th participant. The final codebook in our analysis is available in the artifact link. For the quantitative data obtained from the Likert scale in measuring the level of concern towards impersonation, we conducted a t-test to determine whether there was a significant difference before and after participants viewed the Doppelganger Demo.

B. Impersonation Detection Attempt

In this section, we outline our attempt to detect impersonation commits and users. The process includes two key steps: (1) Crosscheck for impersonation commits, and (2) identify GitHub users engaged in impersonation.

1) Crosscheck for impersonation commits: During the process of impersonating a committer, it becomes challenging to identify the actual contributor behind a commit directly. This is because we cannot reliably determine the true author of a commit. On the commit page, GitHub displays the impersonated user as the commit user and even provides a user link to their profile, providing a very convincing view that the impersonated victim did in fact create the commit. Similarly, the impersonated user is shown as the login author in the API.

The true identity of the committer remains undisclosed in impersonation. However, the commits only occur in the commit history of the real user and not the displayed user. Therefore, to address this issue, we collect the commit history of the repository along with the user's commit history in Section IV-A1. By cross-checking these two sets of data, it becomes possible to identify commits that exist in the repository history but are not reflected in the displayed user history.

2) Identify GitHub users engaging in impersonation: After identifying the commits by crosschecking the repository and user commit histories, we examine the users responsible for these commits. By identifying these users, we collect their entire commit history during the previously discussed period (Section IV-A1). We presume that these users are more likely to make additional impersonation commits due to their past involvement. Consequently, a user is expected to have multiple commit email addresses, as they intentionally disguise their identity by altering the commit email in their configuration. However, we note this is not always the case and may result in false positives as many users legitimately use several different email addresses.

V. RESULTS

In this section, we present the findings derived from interviews and the analysis of GitHub data to address our research questions.

A. To what extent are GitHub developers aware of and concerned about impersonation? How does their perception change before and after witnessing the Doppelganger demo? (RQ1)

Under RQ1, the first subquestion focuses on the current level of awareness and concerns among GitHub developers regarding impersonation (i.e., before they learn about it from our demo). The second subquestion assesses their level of concern after they learn how easily impersonation can occur. Together, these two subquestions offer a comprehensive discussion of participants' views as their understanding of impersonation evolves.

1) The perceived security and privacy issues on GitHub: Before presenting the Doppelganger Demo, we initially inquired about the security and privacy issues participants are aware of on GitHub. At the start of the interview, it became evident that all participants had been involved in collaborative projects. Significantly, all contributors expressed happiness and excitement upon having their commits accepted by open-source projects. They believe their contributions can help more developers and eagerly anticipate having their names listed in influential projects.

TABLE II
PRIVACY AND SECURITY CONCERNS ON GITHUB

Theme	Description	Representative Example	Participant
No concern	The participants have no privacy or security concerns because GitHub is the most renowned open-source project community.	*I don't see any potential risk in using the GitHub. (P3)*	P3, P5, P8, P14, P15
Public key leakage	The participants are worried about the leakage of sensitive information, such as specific API keys, when they release their code to the public.	*Sometimes, you don't even know when your code will have maybe your API key. (P2)*	P2, P4, P7, P11, P16, P17
Personal email leakage	The participants are concerned that their personal email addresses may be exposed through the GitHub API's commit history.	*I have a several email address. And so I will divide into personal or just something for work or for research. I think it's actually okay for me if I use this email address on my GitHub, it means that is it just for my research or for my work. (P9)*	P9, P13
Personal accounts stolen	The participants are concerned about the possibility of their accounts being stolen by others.	*Not really, cause I haven't saw some account was like still stolen by other people. (P6)*	P1, P6
Project content leakage	The participants are concerned that when they work on GitHub with unreleased projects, their code might be leaked.	*If I'm currently working on a project and that project hasn't yet been published, I'll probably maintain the project into a private mode just in case others could see it. (P10)*	P10, P12

Surprisingly, none of the participants was aware of this impersonation issue before we brought it to their attention (excluding P17). P1 believes it is secure after safeguarding their passwords. He opts for a one-time login using SSH instead of HTTPS, allowing them to renew the password with each session. He believes this practice makes it nearly impossible for attackers to steal their accounts.

> When I use GitHub, I usually am logged in, in the command line, or I usually use SSH whenever it's possible. So there's a command that you can run to basically check that you're logged in. And that whenever I cloned repositories, I use the SSH option instead of the HTTPS. option. But I don't really know anything about how someone could impersonate my GitHub account unless they had my password or if they had my phone as well, because I actually have the one-time password configured. (P1)

Instead, participants did express other security concerns. In Table II, we categorized the concerns of the participants into five distinct categories: Public Key Leakage, Personal Email Leakage, Personal Accounts Stolen, Project Content Leakage, and No Concern. 5 participants did not perceive any security issues existing in GitHub. Given GitHub's standing as the most influential open-source community, they have confidence that the platform is comprehensively secured. Among the remaining 11 participants who express security concerns on GitHub, 6 are apprehensive about the possible public keys leakage (e.g., API keys) when releasing code to the public. Additionally, 2 participants expressed individual concerns about project content leakage (P10, P12), personal account theft (P1, P6), and personal email leakage (P9, P13). P9, in particular, acknowledged the potential for email leakage and took proactive measures by creating a dedicated email specifically for GitHub use.

2) Comparing the degree of concern about impersonation before and after the Doppelganger demo: As discussed above, we investigate participants' awareness of security issues as well as impersonation. Then, treating the Doppelganger Demo in Section II as an intervention, we sought to understand their reactions to the demo and their overall perception of impersonation. As described in Section IV-A4, we employ a Likert scale to gauge participants' level of concern regarding impersonation before and after we showed them the Doppelganger Demo. We use a scale of 1 to 10, where 10 reflects the highest level of concern and 1 indicates little concern. In Figure 4, we illustrate their degree of concern, we observe that, except for P14, all participants showed an increase in concern following the demo. The average level of concern increased from **3.9** to **7.3**, and the median value rose from **3** to **7.25**. With a t-test, we found t =-4.24, $p < 0.001$, indicating a significant difference in concern levels before and after the intervention.

Interestingly, this difference also varies based on the participants' experiences as shown in Figure 5. The novice and Expert groups showed a similar degree of concern before the demo (4.1 vs. 3.3), but the Novice group was more worried after the demo (7.9 vs. 5.5). The difference in concern levels was more pronounced between industry and non-industry participants. The average concern level of the industry group before the demo is **2.67**, compared to **4.6** for the non-industry group. After the demo, the average concern levels increase to **6.08** for the industry group and **8.05** for the non-industry group. Though the concern level comparison did not survive the statistical test ($p = 0.06$), industry professionals generally

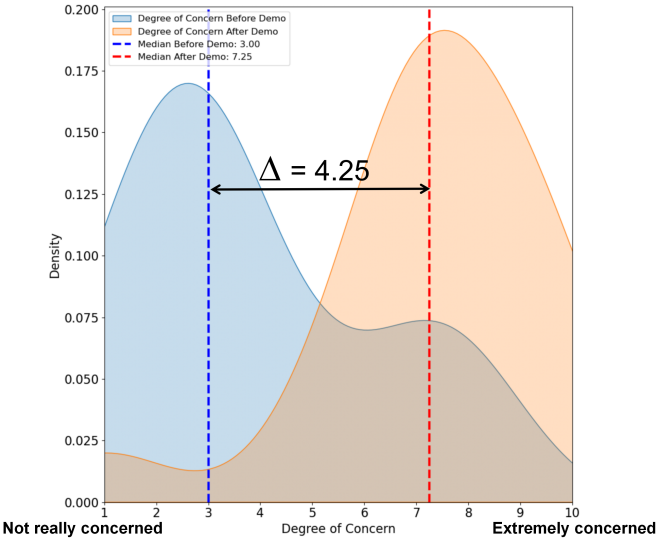

Fig. 4. The distribution of the degree of concern before and after the Doppelganger Demo. The delta indicates the median degree of concern, which saw a significant increase after participants viewed the demo (t=-4.24, $p < 0.001$).

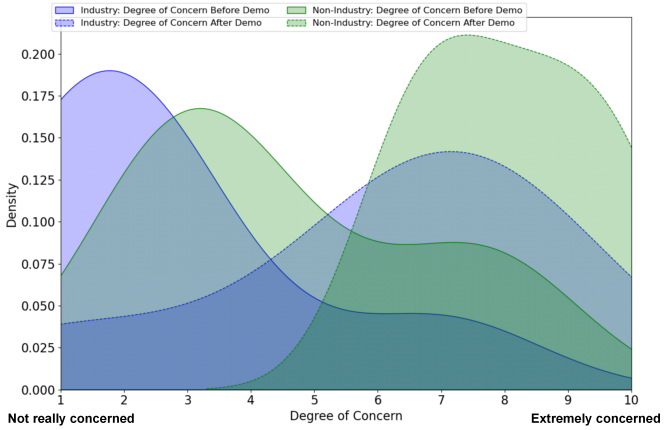

Fig. 5. The comparison between the degree of concern before and after the Doppelganger Demo for industry and non-industry participants. (t=-1.96, $p = 0.06$)

exhibit lower concern towards impersonation. This might be because of the stricter identity verification processes in the industry and the prevalence of proprietary code libraries in large companies, reducing reliance on GitHub, while non-industry individuals face higher risks of impersonation due to less stringent identity verification requirements.

Concern for impersonation before the Doppelganger Demo. Among all participants, 6 expressed the belief that impersonation is uncommon, unlikely to occur, and not a significant concern for them (P1, P3, P9, P11, P12, P14). Their rationale for this perspective is rooted in the belief that faking an identity would require a considerable investment of resources or time, making it an impractical pursuit for the average user. Additionally, they believe that influential users, in particular, typically benefit from robust protection measures for their GitHub accounts. Interestingly, 3 other participants agree that the user identity leak only affects influential developers compared to the normal one (P2, P4, P7). They perceive identity leaks among normal users as unlikely to yield significant risks. Given that normal users typically have fewer contributions to fundamental projects, even if someone were to impersonate them, the potential for causing substantial harm is considered minimal. One participant has claimed if it is easy to pretend to be other the Github is an unsafe open-source community (P6).

Overall, before our demonstration, 12 out of 16 participants expressed a certain level of concern about impersonation, a higher proportion compared to the 11 out of 16 participants who acknowledged security issues on GitHub. Among these, 3 participants believed that impersonation primarily affects influential developers, while another 4 thought it had the potential to diminish the reputation of all users. One participant even expressed the belief that every project is unsafe. The majority of participants promptly view impersonation as a security concern upon acknowledging its potential existence on GitHub, although they may not perceive it as highly severe.

The concern for impersonation after demo After showing our Doppelganger demo to guide participants through the entire impersonation process, we inquire about their attitudes towards impersonation again. Participants' attitudes underwent a significant shift after witnessing our impersonation demo. **7 participants who initially considered impersonation as uncommon or only relevant to influential developers (P2, P4, P5, P6, P7, P8, P11) now perceive impersonation as more harmful than they had initially expected.**

From the participant's view after the Doppelganger demo, 10 participants viewed impersonation as a significant issue due to its ease of reproduction, requiring only 3–4 git commands familiar to every developer. Participants even expressed suspicion about whether we used additional tools in the demonstration (P9). Additionally, participants emphasized that attackers do not even need to know one's password and impersonation is unexpectedly easy.

Also, 4 participants expressed shock after witnessing the demo, emphasizing their surprise that individuals could even access the link to the displayed author (P2, P9, P13, P16). Furthermore, 2 participants suggested that the entire Git system should be enhanced to prevent the occurrence of impersonation (P4, P5). Finally, 2 participants expressed confusion regarding the existence of a serious vulnerability on GitHub, questioning why the identity of every user would be entirely public (P6, P14).

Furthermore, the participants also shared with us their opinions about how impersonation could influence the GitHub community. Table III presents the themes and representative examples from the interview results. 5 participants perceive impersonation as **posing risks to the reliability and safety** of the entire platform. They believe impersonation introduces challenges for the system to accurately identify commit contributions, which forms the foundation of the open-source community. At times, developers can unfairly add other people to the contributor list by using their email, which is not fair to genuine contributors:

TABLE III
THE INFLUENCE TO GITHUB WITH INTERVIEWEE'S OPINION

Theme	Description	Representative Example	Participant
Open-Source-Software Education Integrity	For students, someone might submit low-quality code using the names of those they dislike to sabotage their performance on the project.	*They have your email and that thing will be this whole thing will be true. They can just easily pretend to be you and I'll post some random s***. So I* ***random trash up to the code.*** (P2)	P2
Reduced trust in collaboration	The misuse of GitHub identities causes developers to doubt the suggestions and code contributions of their collaborators.	*I think it's also like you showed it can be misused, where someone could push out a bunch of commits to different repositories, but it's not actually the commits that they're, they're associated with a different identity than the person that's making the commit. So I think there's an opportunity to basically make a coder look bad by pushing bad code using their email. Or you can* ***basically have someone take credit for work that you've done.*** (P1)	P1, P8, P10, P11, P13, P14
Risky reliability and safety	The entire GitHub platform will be affected because it's now difficult to determine the true author of the commit.	*It might affect the integrity of commit attribution and* ***make it challenging to accurately identify and verify computers.*** *Potentially impacting collaboration and reliability of the platform. (P8)*	P6, P8, P9, P10, P16
User Sabotage	For developers, someone could submit low-quality work using another person's identity to diminish their credit in the workplace.	*In my work* ***they can pretend to be me and publish a very bad code on company report*** *and to make me, like, maybe have a bad code quality and they have some of fake news about me like I did badly by showing such fake information and it's fake in history.* (P7)	P7
Project Sabotage	The attackers can impersonate a main contributor of an open-source project and submit malware, which may be easily accepted.	*I think since impersonation may be misused by some like will like your will that* ***attackers that they may pretend to be you and commit something that you don't want to happen to.*** *Some open source project if it is critical like fundamental component or belief system or other fundamental systems and if you were believed to be the programmer and have committed something, probably you will have trouble of it. So yeah, this definitely involves more potential security issues.* (P4)	P2, P3, P4, P12, P15

> Collaborator contributes to this repository and use others, like account, which means others' names will also be a list of the contributors, but they actually didn't do anything for this repository. So I don't think it's very like it's been fair to the contributor itself. (P6)

5 participants expressed concern about **project sabotage**, where attackers could submit pull requests with malicious code to an open-source project by posing as a senior contributor. Typically, project owners tend to place more trust in developers they already know and may not thoroughly review the code contributions. If an attacker successfully impersonates such a trusted person, there is a higher likelihood that malicious code could be accepted and merged into the project. Participants also believe that it could make the entire community less active.

Furthermore, 6 participants believe that impersonation is likely to **reduce trust in collaboration** within the open-source community. In a GitHub project, if contributors can easily impersonate others, determining the true originator of a commit becomes challenging, which undermines the trust in the collaborative process. There is a risk associated with using a credible developer's identity to submit vulnerable code to a repository.

A participant highlights that impersonation could potentially create issues in their school projects as **user sabotage**. In computer science courses, certain assignments require submission via GitHub to the professor. If someone can impersonate others and intentionally submit a code of poor quality, the student being impersonated may receive a lower grade. Typically, students use their school email as their associated email on GitHub, making impersonation even easier. Furthermore, one participant is concerned about the impact of impersonation in the industry. In some teams, the quality of code is used to assess a developer's performance. If someone maliciously impersonates others and publishes low-quality code, their performance evaluation could be unfairly downgraded.

According to the participants' perspectives, impersonation has a diverse range of impacts on GitHub. It diminishes the reliability of the entire platform and hampers collaboration between developers. Furthermore, both students and developers can be adversely affected by the malicious act of impersonating and submitting low-quality code.

> Surprisingly, none of the participants were aware of impersonation before the interview. The majority of them underestimated the severity before the Doppelganger Demo; however, their concern levels increased after walking through the impersonation process.

B. How widely is the current general practice (i.e. commit signing) accepted and used among developers? (RQ2)

In this section, we investigate participants' perspectives on practices like commit signing to verify the user identity asso-

Fig. 6. An Example of a verified commit: Once a GitHub user enables commit signing and uses 'git commit -S -m,' their commits will be marked as verified.

ciated with GitHub commits. Commit signing is a feature on GitHub designed to verify that a commit is made by a specific user[5]. This process entails associating the cryptographic GPG (GNU Privacy Guard) key with the user's git client.

Despite the appearance that commit signing eliminates impersonation, it has several shortcomings in practice. Firstly, as many as 56% of the collected commits in our dataset have not been verified, even within larger and more influential projects. Additionally, commit signing is associated with a private key, but it does not prevent other developers from pretending to be the verified author. The distinction lies in the fact that impersonated commits would lack the "verified" label. In our interview, 3 participants who have used commit singing previously mentioned that commit signing introduces additional steps during the committing process, and generating a GPG key is not convenient for novice programmers or users who infrequently commit with different local machines.

> I sometimes work on like public workspace. So if I want to try to commit something on it, it must may not be verified, right? Or even I'm working on a virtual machine. And each time to open a random one? That how can I get myself verified? I have to maintain like much keys. It's more like too complicated to manipulate these things to configure out of things for each day when I tried to set a quick commit or something. (P12)

We examined the usage of commit signing among all participants. Out of 17 participants, only 6 had a precise understanding of what commit signing is and how to employ it. Among these individuals, 5 had prior experience with commit signing. Interestingly, following our discussion on impersonation, 9 participants who had never used commit signing expressed a willingness to adopt them in the future, despite acknowledging their imperfections, as commit signing is the only current mitigation to impersonation.

> I will use it. I think it's parts like part of the problem but as you see that it can't really deny, really stops impersonate. (P8)

However, among 5 developers who had previously used commit signing, 3 indicated that they would not use it in the future unless repositories require compulsory verification due to the extra effort on key management.

[5]The example comes from https://docs.github.com/en/authentication/managing-commit-signature-verification/signing-commits, as shown in Figure 6

> Over half of the commits in our dataset were not signed. The majority of our participants had not used commit signing before (9/16), and among those who had, some chose not to continue using them due to the effort involved in key management.

C. To what extent can we detect the potential impersonations on GitHub? (RQ3)

Given that many developers view impersonation on GitHub as a significant concern, and recognize that the existing authentication method of commit signing does not fully mitigate this problem, an approach to distinguish between genuine and impersonated commits does not exist. The GitHub mechanism does not guarantee that the commit author is definitively the displayed author. As shown in Figure 7, **during commit merges, the individual who merges multiple commits into the branch is designated as the displayed author for all changes, even if they were not the ones directly editing the code.** Consequently, when primary contributors merge pull requests into the main repository, they are also listed as the displayed author for all commits. Thus, certain instances in pull requests resemble impersonation, making it challenging to discern authentic impersonation from merge commits in a fork to a pull request.

Moreover, we explore the possibility of detecting impersonation by applying the Crosscheck algorithm to find the commit that existed in the repository event but not in the displayed user event in Section IV-B.

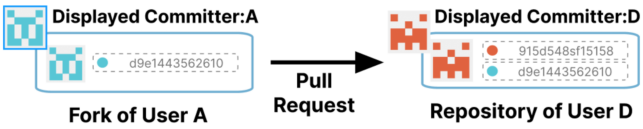

Fig. 7. The illustrated example shows the challenge of distinguishing between impersonated and authentic pull requests.

As outlined in Section IV-A1, we collect commits from the highest-starred repositories by searching random keys. Out of our 9,810 repositories on GitHub, we have accumulated a total of 12,577,332 commits and have documented the commit history of 172,587 users. Among these users, 5,447 individuals have made commits using more than one email address, comprising approximately 3.16% of the total user count. We have uncovered several interesting findings within the total of 131,184 suspicious commits. Notably, 16,420 commits originate from large companies (e.g., Google, Meta), 4,665 commits come from organizational emails, and 1,666 commits are associated with educational emails. Additionally, 120 commits are linked to government emails. These results indicate a correlation between organizational collaboration behavior and impersonation.

The Git mechanism for displaying commit authors is the same for both authentic pull requests and impersonation. Thus, there is no approach yet to accurately and precisely detect the impersonation behavior.

D. How can GitHub users be safeguarded against impersonation from user perceptions? (RQ4)

As previously discussed, impersonation may have a broader impact than initially perceived by GitHub users and there is currently a lack of effective detection methodology for GitHub impersonation. Therefore, addressing how to safeguard GitHub users and mitigate the effects of identity leaks becomes a crucial concern. In this section, we summarize participants' opinions on this matter, discussing potential measures that can be taken to mitigate impersonation. The opinions from the industry and non-industry groups do not differ. However, the Expert and Novice group focuses on different types of solutions. We present the opinions from both groups to provide a comprehensive view of how GitHub users would do to safeguard themselves on impersonation, considering their different levels of experience.

Participants in the Expert group, who are familiar with various GitHub functions, offered suggestions based on existing GitHub commit features:

- **Default vigilant mode** The Vigilant mode in GitHub can alert users to all commits linked to their account (P13). However, the default setting for Vigilant mode is off. When Vigilant mode is activated, every developer can receive notifications if their identity has been used elsewhere.
- **Compulsory commit signing policy** GitHub could mandate commit signing as a default setting for all commits. This would ensure that all commits are authenticated, preventing users from using other people's email addresses (P13, P14, P16). However, implementing this approach necessitates developers to generate keys on all their local machines.

Participants in the Novice group offered suggestions that are innovative and extend beyond existing GitHub functions:

- **Use the public key in commit signing** The participants advocate for GitHub to enhance the current commit signing feature by integrating a public key (P4, P10). This improvement aims to deny access without introducing additional verification steps, specifically when other individuals attempt to use the email of a verified user.
- **Use specific keys in commit message** Another participant proposes that committers leverage a specific commit message, established by the repository owner, as a way of verifying identity (P1, P12). In this approach, the repository owner gains the ability to confirm the true identity of the contributor using the designated commit message, posing a challenge for potential attackers attempting to impersonate an existing collaborator.
- **Call for GitHub to raise user's security awareness** A participant expresses the desire for GitHub to officially disclose potential risk vulnerabilities, including impersonation (P5).

Although some of the perceptions from participants may not be rigorously verified, they can still provide insights into features that could help detect impersonation. Participants in the Novice group proposed more inspiring mitigation strategies that are not currently present in GitHub's mechanisms. Conversely, the Expert group provided more technical suggestions, focusing on improvements to existing functions. In the discussion section, we elaborate on strategies to mitigate the influence of these factors and explore reasons why GitHub has not addressed this issue.

VI. Discussion

In our discussion, we delved deeper into the impact of impersonation on OSS, GitHub's official treatment of impersonation since its awareness of its existence several years ago, and strategies for mitigating this behavior without disrupting the original function of OSS.

The impact of impersonation on OSS Two years ago, there was a noteworthy (albeit humorous) incident involving Linus Torvalds' GitHub identity, where it was claimed that he would delete the entire Linux system [44]). This real-world example illustrates how impersonation can be used to deceive even renowned developers. Besides impersonating influential developers to make headlines, impersonation can also happen discreetly across a broader spectrum of repositories. From injecting malware into a project under a main developer's identity to pushing low-quality code using other individuals' accounts to undermine their reputation, impersonation could potentially have a substantial impact, and detecting the real committer within the Git mechanism is challenging.

GitHub's stance on impersonation. GitHub has officially acknowledged the presence of impersonation and has included impersonation as prohibited content or activity in its site policy [45]. *The Git configuration is flexible, allowing any developers to use other people's identities because GitHub considers it convenient for several contributors to share credit for their commits* (P17).

However, existing mechanisms are unable to definitively determine the true user behind a commit. Technically, identifying violations of user identity can only depend on the assumption that malicious commits are not made by any legitimate developers on GitHub.

Ways to mitigate impersonation While commit signing may appear to offer a solution for the impersonation issue, 60% of our participants who have used commit signing before stated that they would not consider using it unless required by a project. Impersonation may be more problematic among influential and highly active developers. However, these individuals often face challenges in managing keys for signing commits, particularly because they may use several local or remote machines. Additionally, implementing email authentication for every commit presents challenges, especially

when multiple contributors share credit for a commit or pull request that entails multiple email addresses.

Presently, the push event of a pull request mirrors impersonation through the use of other people's email addresses — a pull request may contain commits made by others, a natural consequence of the git architecture. It is essential to prioritize the detectability of impersonation rather than manually policing for such events. For instance, in the GitHub API, instead of solely displaying information based on the Git configuration from the committer, it can include the ID for each Git client to identify the individuals who push the commit.

Implications for future work: Our study represents the first investigation into the occurrence of impersonation and developers' attitudes towards it. Currently, impersonation behaviors are not automatically detected. We aim for our study to provide insights for future research. For developers, we hope to raise awareness about identity verification issues in the open-source community and practical strategies to mitigate risks within the current development ecosystem. Researchers need to further explore how impersonation influences open-source software (OSS) and potential mechanisms within Git to mitigate these issues while maintaining flexibility.

VII. Limitations

As the first work on investigating impersonation on GitHub, our study has multiple limitations. In our qualitative study, we encountered potential biases that may affect the validity and generalizability of our findings. We discuss each below.

Recruitment Biases: In our recruitment process, we select developers from repositories based on our selection strategy, which involves choosing from the top 1,000 most-starred repositories across ten searches using randomly generated two-letter strings. Participants are selected only if they have committed to one of these repositories. This approach introduces bias because the majority of participants are not relatively novice programmers, as they have already committed to these repositories. Our methodology of using the GitHub Search API with random two-letter strings may have yielded smaller repositories with few stars or whose contributors were more novices compared to other individuals in the participant cohort. Overall, our participant pool may not represent the whole population in the GitHub community, which might lead to issues with generalizing our findings.

Response Biases: A social desirability effect may occur during the interview, especially as we introduce the Doppelganger demo. Consequently, participants may feel compelled to express their concerns about impersonation as a result of our mentioning it. It could be possible that the impact on participants' views of impersonation was impacted by the fact that we were confronting them about it.

Furthermore, our results and discussion on potential safeguarding mechanisms against impersonation are based on our participants' views, who might not have a comprehensive background on effective defense strategies for the open source community, which might largely limit the insights for future work on impersonation detection and defense. We hope our current report from GitHub users can initiate more future work to further explore solutions to prevent GitHub impersonation, and indicate candidate designs of such defense that might be easily adopted by GitHub users.

VIII. Conclusion

In our study, we conducted the first study to investigate impersonation in GitHub. This security issue allows users to easily pretend to be other developers by committing with just a few Git commands to configure a different email.

We conducted interviews with developers in GitHub. The interviews revealed that most of the participants were unaware of the impersonation issue before our investigation. When mentioning the possibility of impersonation, over half of the participants initially perceived it as not having significant security issues. However, after showing them a demonstration of impersonation, most of the participants recognized it as a serious vulnerability. This highlights that the severity of impersonation is often underestimated or even neglected by GitHub users.

We also examined the adoption of the current mitigation practice, known as commit signing, to mark commits as verified using cryptography. Our evaluation of its usage in our dataset revealed that 56% of commits do not have verification. Moreover, during our interviews, fewer than one-third of the participants were familiar with commit signing before our discussions. Over half of the participants who had previously used it did not plan to use it in the future unless required, and they highlighted several shortcomings associated with it.

Additionally, we determined that filtering out malicious impersonation from multi-email commits is not currently feasible. The impersonation behavior remains undetectable due to its similarity to pull requests. We developed an algorithm to detect potential identity leaks, analyzing 12.5 million commits across 9810 public repositories. Our findings revealed that 0.6% of commits and 3.16% of users were implicated in mismatches between the displayed author and code author. All of these instances have the potential to be cases of impersonation, but we cannot automatically determine whether they are true instances of impersonation or legitimate coincidences of the git architecture.

In summary, impersonation on GitHub has a broader and more serious impact than anticipated according to our participants. We hope that GitHub can officially address and improve the identified issues, whether through enhancements to signing commits or the addition of email configuration authentication.

IX. Acknowledgment

This work was supported by the National Science Foundation (NSF) under grant CCF-2211429. Partial support was provided by the NSA Science of Security program, the ARPA-H program, and NSF award 2312057. The authors thank these organizations for their support.

REFERENCES

[1] R. Naraine, "Open-source proftpd hacked, backdoor planted in source code," 2010.

[2] A. Zagalsky, J. Feliciano, M.-A. Storey, Y. Zhao, and W. Wang, "The emergence of github as a collaborative platform for education," in *Proceedings of the 18th ACM conference on computer supported cooperative work & social computing*, 2015, pp. 1906–1917.

[3] A. Lima, L. Rossi, and M. Musolesi, "Coding together at scale: Github as a collaborative social network," in *Proceedings of the international AAAI conference on web and social media*, vol. 8, no. 1, 2014, pp. 295–304.

[4] Y. Hu, S. Wang, Y. Ren, and K.-K. R. Choo, "User influence analysis for github developer social networks," *Expert Systems with Applications*, vol. 108, pp. 108–118, 2018.

[5] G. B. Alves, M. A. Brandão, D. M. Santana, A. P. C. da Silva, and M. M. Moro, "The strength of social coding collaboration on github," pp. 247–252, 2016.

[6] A. Begel, J. Bosch, and M. Storey, "Social networking meets software development: Perspectives from github, msdn, stack exchange, and topcoder," *IEEE Software*, vol. 30, pp. 52–66, 2013.

[7] Customizing git - git configuration. [Online]. Available: https://git-scm.com/book/en/v2/Customizing-Git-Git-Configuration

[8] Izoologic. (2022) Attackers target developers by impersonating github commits metadata. Accessed: February 21, 2024. [Online]. Available: https://izoologic.com/region/us/attackers-target-developers-by-impersonating-github-commits-metadata/

[9] GitHub. (2023) Security alert: Social engineering campaign targets technology industry employees. [Online]. Available: https://github.blog/2023-07-18-security-alert-social-engineering-//campaign-targets-technology-industry-employees/

[10] (August 2023) Impersonation attacks target github developers. University of California, San Francisco (UCSF). [Online]. Available: https://it.ucsf.edu/aug-2023-impersonation-attacks-target-github-developers

[11] Y. Acar, C. Stransky, D. Wermke, M. L. Mazurek, and S. Fahl, "Security developer studies with github users: Exploring a convenience sample," in *Thirteenth Symposium on Usable Privacy and Security*, 2017, pp. 81–95.

[12] R. Feng, Z. Yan, S. Peng, and Y. Zhang, "Automated detection of password leakage from public github repositories," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 175–186.

[13] B. Lazarine, S. Samtani, M. Patton, H. Zhu, S. Ullman, B. Ampel, and H. Chen, "Identifying vulnerable github repositories and users in scientific cyberinfrastructure: An unsupervised graph embedding approach," in *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 2020, pp. 1–6.

[14] H. Afzali, S. Torres-Arias, R. Curtmola, and J. Cappos, "le-git-imate: Towards verifiable web-based git repositories," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, 2018, pp. 469–482.

[15] S. Torres-Arias, A. K. Ammula, R. Curtmola, and J. Cappos, "On omitting commits and committing omissions: Preventing git metadata tampering that (re) introduces software vulnerabilities." in *USENIX Security Symposium*, 2016, pp. 379–395.

[16] Github impersonation - acceptable use policies. [Online]. Available: https://docs.github.com/en/site-policy/acceptable-use-policies/github-impersonation

[17] A. Bosu, J. C. Carver, M. Hafiz, P. Hilley, and D. Janni, "Identifying the characteristics of vulnerable code changes: An empirical study," in *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 257–268.

[18] A. Meneely, H. Srinivasan, A. Musa, A. R. Tejeda, M. Mokary, and B. Spates, "When a patch goes bad: Exploring the properties of vulnerability-contributing commits," in *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE, 2013, pp. 65–74.

[19] Q. Wu and K. Lu, "On the feasibility of stealthily introducing vulnerabilities in open-source software via hypocrite commits," *Proc. Oakland*, 2021.

[20] P. Ladisa, H. Plate, M. Martinez, and O. Barais, "Sok: Taxonomy of attacks on open-source software supply chains," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 1509–1526.

[21] S. Amreen, B. Bichescu, R. Bradley, T. Dey, Y. Ma, A. Mockus, S. Mousavi, and R. Zaretzki, "A Methodology for Measuring FLOSS Ecosystems," in *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability*, B. Fitzgerald, A. Mockus, and M. Zhou, Eds. Singapore: Springer Singapore, 2019, pp. 1–29. [Online]. Available: $http://link.springer.com/10.1007/978-981-13-7099-1_1$

[22] M. Zahedi, M. Ali Babar, and C. Treude, "An empirical study of security issues posted in open source projects," in *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018.

[23] D. Pletea, B. Vasilescu, and A. Serebrenik, "Security and emotion: sentiment analysis of security discussions on GitHub," in *Proceedings of the 11th Working Conference on Mining Software Repositories*. Hyderabad India: ACM, May 2014, pp. 348–351. [Online]. Available: https://dl.acm.org/doi/10.1145/2597073.2597117

[24] M. Morton, J. Werner, P. Kintis, K. Snow, M. Antonakakis, M. Polychronakis, and F. Monrose, "Security Risks in Asynchronous Web Servers: When Performance Optimizations Amplify the Impact of Data-Oriented Attacks," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, Apr. 2018, pp. 167–182.

[25] M. Ohm, H. Plate, A. Sykosch, and M. Meier, "Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks," *Detection of Intrusions and Malware, and Vulnerability Assessment*, vol. 12223, pp. 23–43, Jun. 2020.

[26] N. Alexopoulos, M. Brack, J. P. Wagner, T. Grube, and M. Mühlhäuser, "How long do vulnerabilities live in the code? a {Large-Scale} empirical measurement study on {FOSS} vulnerability lifetimes," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 359–376.

[27] "An Analysis of Open-source Automated Threat Modeling Tools and Their Extensibility from Security into Privacy," https://www.usenix.org/publications/loginonline/analysis-open-source-automated-threat-modeling-tools-and-their, Feb. 2022.

[28] "Signing commits." [Online]. Available: https://ghdocs-prod.azurewebsites.net/en/authentication/managing-commit-signature-verification/signing-commits

[29] H. Afzali, S. Torres-Arias, R. Curtmola, and J. Cappos, "le-git-imate: Towards Verifiable Web-based Git Repositories," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ser. ASIACCS '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 469–482. [Online]. Available: https://dl.acm.org/doi/10.1145/3196494.3196523

[30] T. Fry, T. Dey, A. Karnauch, and A. Mockus, "A Dataset and an Approach for Identity Resolution of 38 Million Author IDs extracted from 2B Git Commits," in *Proceedings of the 17th International Conference on Mining Software Repositories*, ser. MSR '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 518–522. [Online]. Available: https://dl.acm.org/doi/10.1145/3379597.3387500

[31] I. Richter, F. Raith, and M. Weber, "Problems in agile global software engineering projects especially within traditionally organised corporations: [an exploratory semi-structured interview study]," in *Proceedings of the Ninth International C* Conference on Computer Science & Software Engineering*, 2016, pp. 33–43.

[32] "GitHub REST API Documentation," https://docs.github.com/en/rest?apiVersion=2022-11-28, accessed: December 14, 2024.

[33] Github, "Use the rest api to search for specific items on github," *GitHub Docs*, 2022. [Online]. Available: https://docs.github.com/en/rest/search?apiVersion=2022-11-28

[34] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 155–165.

[35] Github, "Use the rest api to check your current rate limit status," *GitHub Docs*, 2022. [Online]. Available: https://docs.github.com/en/rest/rate-limit?apiVersion=2022-11-28#about-rate-limits

[36] ——, "List repository contributors," *GitHub Docs*, 2022. [Online]. Available: https://docs.github.com/en/rest/repos/repos?apiVersion=2022-11-28#list-repository-contributors

[37] ——, "Use the rest api to interact with github events," *GitHub Docs*, 2022. [Online]. Available: https://docs.github.com/en/rest/activity/events?apiVersion=2022-11-28

[38] ——, "Use the rest api to interact with commits," *GitHub Docs*, 2022. [Online]. Available: https://docs.github.com/en/rest/commits/commits?apiVersion=2022-11-28

[39] M. Endres, K. Boehnke, and W. Weimer, "Hashing it out: A survey of programmers' cannabis usage, perception, and motivation," in *Proceedings of the 44th International Conference on Software Engineering*, 2022, pp. 1107–1119.
[40] Y. Huang, D. Ford, and T. Zimmermann, "Leaving my fingerprints: Motivations and challenges of contributing to oss for social good," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1020–1032.
[41] M. Wahl, J. Krüger, and J. Frommer, "Users' sense-making of an affective intervention in human-computer interaction," in *Human-Computer Interaction. Novel User Experiences: 18th International Conference, HCI International 2016, Toronto, ON, Canada, July 17-22, 2016. Proceedings, Part III 18*. Springer, 2016, pp. 71–79.
[42] J. Dawes, "Do data characteristics change according to the number of scale points used? an experiment using 5-point, 7-point and 10-point scales," *International journal of market research*, vol. 50, no. 1, pp. 61–104, 2008.
[43] J. Fereday and E. Muir-Cochrane, "Demonstrating rigor using thematic analysis: A hybrid approach of inductive and deductive coding and theme development," *International journal of qualitative methods*, vol. 5, no. 1, pp. 80–92, 2006.
[44] Sobyte. (2022) Linus play a trick of github vulnerability. [Online]. Available: https://www.sobyte.net/post/2022-01/linus-play-a-trick-of-github-vulnerability/
[45] GitHub. (n.d.) Github impersonation. Accessed: February 21, 2024. [Online]. Available: https://docs.github.com/en/site-policy/acceptable-use-policies/github-impersonation