Poster: Committed by Accident – Prevention and Remediation Strategies Against Secret Leakage

Alexander Krause*, Jan H. Klemmer†, Nicolas Huaman*,
Dominik Wermke*, Yasemin Acar‡, Sascha Fahl*
*CISPA Helmholtz Center for Information Security, Germany, {first.last}@cispa.de
†Leibniz University Hannover, Germany, klemmer@sec.uni-hannover.de
‡George Washington University, USA, acar@gwu.edu

***Abstract*—Version control systems for source code, such as Git, are key tools in modern software development environments. Many open-source projects use online services such as GitHub or GitLab. Previous work and news articles illustrate that developers tend to commit code secrets such as private encryption keys, passwords, or API keys accidentally. However, making secrets available to the public Internet might have disastrous consequences, such as leaving systems vulnerable to attacks. In a mixed-methods study, we surveyed 109 developers, including 50 freelancers from Upwork and 59 developers from GitHub, with a focus on their strategies for secret leakage prevention and their secret leakage experiences. We also analyzed 100 online guidelines for secret leakage prevention and remediation. We find that 30.3% of our participants have encountered secret leakage in the past, and the online guidelines we analyzed do not sufficiently address secret leakage prevention and remediation. We conclude with recommendations for developers and an outlook on this research.**

I. Introduction

Version control systems (VCSs) are an essential technology for collaborative software development in the 21st century, with Git being the most commonly used tool [1]. Public online code repository platforms (e. g., GitHub and GitLab) allow for easy sharing, reviewing, and contributing to software projects. In modern development pipelines, software is commonly directly built, tested, and deployed from these code repositories. To deploy software on server infrastructure, automate interactions with third-party services, or handle authentication, developers need to provide secrets, e. g., credentials, authentication tokens, or secret keys. However, these secrets must be protected from accidentally leaking into the public codebase. This is no straightforward task, as recent work by Meli et al. has shown that on GitHub thousands of automatically detectable secrets are leaked every day [2]. GitLab also acknowledges this problem, e. g., by stating: "A recurring problem [...] is that people may accidentally commit secrets to their remote Git repositories." [3].

The actual impact of a leaked secret depends on the type of secret. In some cases, a leak can be highly critical, as in the case of SolarWinds: A trivial password for their update servers was publicly pushed to GitHub in 2017. It allowed attackers to modify production code on SolarWinds' build server and distribute vulnerable patches to up to 18,000 customers [4]. The ongoing attack was only discovered two years later by security researcher Vinoth Kumar [5].

In this work, we investigate approaches to remediate and prevent code secret leakage with a mixed-method approach to answer the following research questions:

RQ1:. *What code secret management approaches are available online?*

RQ2:. *Which approaches to code secret management are developers using in real-world development?*

RQ3:. *What are developers' experiences with secret management approaches?*

II. Methodology

We used a mixed-methods approach to answer the research questions. First, we analyzed 100 online guides for secret leakage prevention and remediation approaches. Subsequently, we surveyed 109 developers from Upwork and GitHub about their experiences with code secret leakage.

A. Guide Analysis

To address **RQ1**, we analyzed code secret management approaches present in online resources and guides. Therefore, we surveyed 18 undergrad CS students to define a search term for code secret leakage prevention and remediation each, resulting in 17 distinct queries. We then used *Google Search* to perform an online search for each query. In compliance with past research [6], we only considered the top-five Google search results. From those, we randomly selected 50 documents each for prevention and remediation. In total, we reviewed 100 different guides. Three researchers identified prevention and remediation approaches using *iterative categorization* [7].

B. Survey

We surveyed a diverse set of developers using an online questionnaire to answer **RQ2** and **RQ3**. We iteratively developed the questionnaire and tested it with four usable security researchers in cognitive walkthroughs. Finally, we piloted the survey with 11 participants and iteratively improved the questions. The survey structure has several parts that queried participants on the following topics subsequently: (1) usage of source code management and VCS tools, (2) experience in handling secrets, (3) threat model for secrets, (4) secret leakage remediation, (5) secret leakage prevention approaches, and (6) demographics.

Table I: Demographics of participants from both Upwork and GitHub as well as both combined.

	Upwork	GitHub	Combined
$n =$	50	59	109
Gender:			
Male	86.0%	88.1%	87.2%
Female	10.0%	1.7%	5.5%
Non-Binary	0.0%	6.8%	3.7%
Age [years]:			
Median	29.0	33.0	30.0
Mean	31.3	34.9	33.2
Std. Dev.	8.6	12.7	11.1
Country of Residence:			
U.S.	2.0%	32.2%	18.3%
India	20.0%	3.4%	11.0%
Germany	0.0%	18.6%	10.1%
Pakistan	14.0%	3.4%	8.3%
Other[1]	60.0%	40.7%	49.5%
Development Experience:			
$<$ 1 year	6.0%	1.7%	3.7%
1–2 years	16.0%	3.4%	9.2%
2–5 years	42.0%	25.4%	33.0%
$>$ 5 years	36.0%	69.5%	54.1%

[1] Other countries are those that occurred overall $< 4\%$.

For the survey, we hired 50 freelance developers on Upwork and 59 developers on GitHub. We report the participant demographics in Table I. Overall, the demographics are comparable to the latest Stack Overflow developer survey [1] in terms of gender, age, top-3 countries, and education.

Our institution's ethical review board (ERB) approved the study. Further, we adhered to the strict German privacy laws and the General Data Protection Regulation (GDPR). The Upworkers were rewarded with $25 for their participation.

III. Findings

In the following, we highlight selected (preliminary) results.

Code Secret Leakage Prevelance. We found that 30.3% of our participants experienced code secret leakage themselves in the past. Further, 38.5% know others who experienced secret leakage in the past, as depicted in Figure 1.

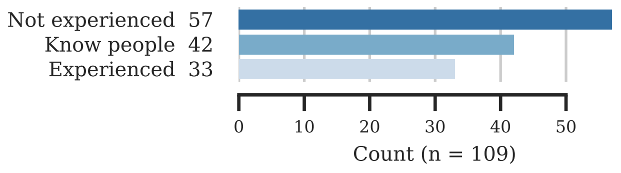

Figure 1: Developers reported experience on code secret leakage in the past or know people who did. We allowed multiple answers.

Code Secret Management Approaches. In total, we discovered 18 distinct approaches developers used to prevent and remediate code secret leakage. The attached poster illustrates the approaches in a table, with additional information on the prevalence of the approaches in both the survey and the analyzed guides. The guide analysis revealed that only 29% of the analyzed resources were about preventing or remediating code secret leakage. In addition, we found that resources vary widely in completeness and level of detail, which can make it difficult for developers to find the right guidance.

IV. Recommendations for Developers

A. Prevention

We suggest that developers use a combination of different approaches to decrease the likelihood of code secret leakage. First, developers should *externalize secrets* and *block secrets* to prevent committing a code secret to publicly available source code repositories. These approaches can be strengthened by also applying *monitoring*, especially in a pre-commit approach before pushing a commit to the server. Sometimes, developers need to share code secrets through the repository with others. In that particular case, developers should use *encrypted secrets*. This approach can also be used preventive, as an attacker cannot access accidentally published encrypted secrets without the encryption key.

B. Remediation

Typical steps that should always be taken to effectively remediate leaked code secrets are *renewing or revoking the secret*, *analyze leak* and using those results revising *access management*. We also consider it essential to *notify the concerned roles* for legal and ethical reasons, if not to get the appropriate help from security and privacy experts. We consider the previous steps necessary because they handle all consequences of a secret leak. *Removal from source code* and *cleanup VCS history* are important steps. However, they cannot save a publicly leaked secret, as it might already be disclosed to crawlers or other users [2]. This empathizes the need to *renew or revoke secret*s that have leaked in public spaces.

V. Outlook

Furthermore, we plan to reveal potential factors impacting code secret leakage, analyze official remediation guidelines of service providers, and give recommendations for service providers on improving online guides and provision and extension of secret scanning. The results also need to be discussed in terms of usability and adoption. Future work could improve the understanding of the general causes of code secret leakage.

References

[1] Stack Overflow, "Stack Overflow Developer Survey 2021," 2021, https://insights.stackoverflow.com/survey/2021 (visited on 02/01/2022).
[2] M. Meli, M. R. McNiece, and B. Reaves, "How bad can it git? characterizing secret leakage in public github repositories." in *NDSS*, 2019.
[3] GitLab Inc., "Secret Detection," https://docs.gitlab.com/ee/user/application_security/secret_detection/ (visited on 02/01/2022).
[4] I. Jibilian and K. Canales, "The US is readying sanctions against Russia over the SolarWinds cyber attack." https://www.businessinsider.com/solarwinds-hack-explained-government-agencies-cyber-security-2020-12 (visited on 02/01/2022).
[5] L. Williams, "The people who live in glass houses are happy the stones weren't thrown at them [from the editors]," *IEEE Security & Privacy*, vol. 19, no. 3, pp. 4–7, 2021.
[6] B. Pan, H. Hembrooke, T. Joachims, L. Lorigo, G. Gay, and L. Granka, "In google we trust: Users' decisions on rank, position, and relevance," *Journal of computer-mediated communication*, vol. 12, no. 3, pp. 801–823, 2007.
[7] J. Neale, "Iterative categorization (ic): a systematic technique for analysing qualitative data," *Addiction*, vol. 111, no. 6, pp. 1096–1106, 2016.