

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


Alexander Krause ^C

Jan H. Klemmer ^{*}

Nicolas Huaman ^{*}

Dominik Wermke^C

Yasemin Acar ^{†, ‡}

Sascha Fahl ^C

^C*CISPA Helmholtz Center for Information Security, Germany,*

{alexander.krause, dominik.wermke, sascha.fahl}@cispa.de

^{*}*Leibniz University Hannover, Germany, {klemmer, huaman}@sec.uni-hannover.de*

[†]*Paderborn University, Germany, yasemin.acar@uni-paderborn.de*

[‡]*The George Washington University, USA*

1 Introduction

Version control systems (VCSs) are an essential technology for collaborative software development. Git [1], a fundamental tool to orchestrate collaborative development, has been voted as the most common tool in the recent Stack Overflow Developer Survey [2] with 93.4% of participants specifying to use this tool in their development workflow. Git-based code repository platforms (e. g., GitHub [3] and GitLab [4]) aim to ease sharing, reviewing, and contributing to software projects. In modern development pipelines, software is commonly directly built, tested, and deployed within and 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 encryption keys. However, these secrets must be protected from being leaked accidentally into the public codebase. Unfortunately, this is no straightforward task. Recent work by Meli et al. [5] found that on GitHub, the most popular code sharing platform,¹ thousands of automatically detectable secrets are leaked daily.

A leaked secret can have a significant impact depending on the type of secret and how long it takes for the secret owner to revoke it after noticing its leak. In the case of Toyota, a hard-coded credential for accessing a data server was publicly pushed to GitHub in 2017. It allowed attackers to control the Toyota T-Connect accounts for 296,019 customers [7]. After more than five years, Toyota invalidated the key. Such a long time could mean multiple malicious actors have already gained access. GitHub recently leaked a private SSH host key of their production Git servers in a public repository. This incident illustrates the complexity of secret management, even for large companies experienced with code secrets and public source code repositories [8].

There is anecdotal knowledge [9, 10, 11, 12] on secret leaks through source code repositories. However, there has been little prior research trying to understand better the reasons

¹According to the Tranco list [6] generated on January 5, 2022, available at <https://tranco-list.eu/list/Q674>.

for and experiences with code secret leakage in source code repositories. To address this gap, we investigate the following research questions:

RQ1. *How widespread is code secret leakage among developers?* Leaking secrets and access tokens in source code poses a potentially serious security threat. We asked 109 developers how often they encountered secret leaks in the past.

RQ2. *What are secret leakage prevention approaches, and what are developers experiences?* Depending on the scenario and context, prevention approaches can differ widely. We surveyed and interviewed developers to reveal prevention approaches and the experiences, challenges, and needs that developers have when using them.

RQ3. *What are developers' experiences with code secret leakage incidents?* Little is known about developers' experiences when remediating code secret leaks. We interviewed developers on their latest and most impactful secret leaks to learn from their experiences, how they recognized a leak, and their consequences.

RQ4. *What are developers' experiences with code secret remediation techniques and tools?* Remediating code secret leakage can be challenging. We examine deployed remediation approaches, developers' experiences with these approaches, and their requirements for approaches.

2 Methodology

In a mixed-methods study, we surveyed 109 developers with version control system experience. Additionally, we conducted 14 in-depth semi-structured interviews with developers who experienced secret leakage in the past. Our institution's ethical review board (ERB) approved both studies. Further, we adhered to the strict German privacy laws and the General Data Protection Regulation (GDPR).² We report the participant demographics in Table 1. Overall, the demographics are

²To allow a full replication of our research as well as meta-research, we provide a replication package at <https://prismatic-meerkat-6f3fb6.netlify.app>.

comparable to the latest Stack Overflow developer survey [2] in terms of gender, age, top-3 countries, and education.

Survey. We surveyed a diverse set of developers using an online questionnaire to answer **RQ1** and **RQ2**. 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.

For the survey, we recruited 50 freelance developers on Upwork and 59 developers on GitHub.

Interviews. To answer **RQ3** and **RQ4**, we decided to only interview developers who experienced secret leakage and therefore can report on remediation and past incidents from GitHub. This was the only eligibility criteria to participate in an interview.

Analysis and Coding. We used an iterative open-coding approach to analyze all interview transcripts and survey responses [13, 14, 15]. First, two researchers developed an initial codebook. Afterward, the two researchers developing the codebook coded the responses in multiple rounds. After each iteration, they resolved conflicts by consensus discussion or by introducing new sub-codes. We continued iterative coding until no new codes and themes emerged [16, 17].

3 Findings

In the following, we highlight selected results.

Widespread of Code Secret Leakage. We found that 30.3% of our respondents 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.

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 the survey, including tools our survey respondents reported on.

Code Secret Leakage Incidents. Our interview participants experienced code secret leaks in various places. Most participants reported a secret leak through public source code repositories, including GitHub and GitLab.

The type and frequency of secret leaks vastly varied between participants. All participants reported secret leaks through hard-coded information in source code or configuration files of a project. Some interviewees experienced leaks through accidental commits of configuration files that included API keys, tokens, or login credentials.

The impact of code secret leaks highly depends on their detection. In particular, the time span until leaks are detected is important. Most of our participants reported that GitHub notified them in case of a secret leak so that they could respond quickly. In two cases, GitHub even triggered the revocation of the leaked secrets.

We observed two different types of consequences caused by code secret leakage. Consequences may directly affect the company or software team that is responsible for the leak. However, we also saw consequences for external stakeholders, such as clients of the developed software or the customers of that client. Most of our participants reported, that the secret leak caused additional workload for the team. This included the investigation of the leak, as well as the remediation process.

4 Recommendations for Developers

Finally, we present recommendations based on our findings.

Prevention. We suggest using a combination of different approaches to decrease the likelihood of code secret leakage. First, developers should *externalize secrets*, e. g., using environment variables or tools like vaults and secret managers, and *block secrets* from being added to repositories using e. g., `.gitignore` files. These approaches can both prevent an accidental commit and push of a code secret to publicly available source code repositories. In addition, security can be strengthened by also applying *monitoring*, e. g., using secret scanners, especially as a pre-commit approach, so that potential code secrets can be detected before pushing a commit to a server. Sometimes, developers need to share code secrets through the repository with others. In that case, developers should use *encrypted secrets*, so unauthorized third parties cannot access them. This can be automated through encryption workflows using tools like `git-secret`, `SOPS` and `GPP`.

Remediation. Typical steps that should always be taken to effectively remediate code secret leakage are to *renew or revoke secrets* that leaked to prevent further misuse of affected services, *analyze leaks* to identify the root causes of the leak, and *revise the access management* using those results, e. g., apply more restrictive access management if needed. We also consider it essential to *notify the concerned roles* (e. g., management, security team, customers) for legal and ethical reasons, if not to get the appropriate help from security and privacy experts. Overall, we consider the above steps necessary because these steps will handle all consequences of a secret leak. The other approaches (cf. table in the poster) can be used to compliment the essential ones, and should be considered as a second step depending on the situation. *Removal from source code* and *cleaning up VCS history* are such additional steps. However, these alone are not sufficient, as the risk of archiving of public websites emphasizes the need to *renew or revoke secrets* that have leaked in public spaces. *Server Operations* and *systemic consequences* (e. g., introducing new processes) depend heavily on company policies, the type of leak and how well leakage damage can be prevented when developers can just *renew or revoke secrets* that have leaked. If developers or companies have remediation approaches prepared, we suggest verifying their approaches will work in case of a leak.

References

- [1] Git Project. *Git*. <https://git-scm.com/> (visited on 02/02/2023).
- [2] Stack Overflow. *Stack Overflow Developer Survey 2022*. <https://survey.stackoverflow.co/2022/> (visited on 08/08/2022). 2022.
- [3] GitHub Inc. *GitHub*. <https://github.com/>.
- [4] GitLab Inc. *GitLab*. <https://gitlab.com/>.
- [5] Michael Meli, Matthew R McNiece, and Bradley Reaves. “How Bad Can It Get? Characterizing Secret Leakage in Public GitHub Repositories.” In: *NDSS*. 2019.
- [6] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. “Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation”. In: *Proc. 26th Network and Distributed System Security Symposium (NDSS’19)*. 2019.
- [7] Dwayne McDaniel. *Toyota Suffered a Data Breach by Accidentally Exposing A Secret Key Publicly On GitHub*. <https://blog.gitguardian.com/toyota-accidentally-exposed-a-secret-key-publicly-on-github-for-five-years/> (visited on 02/02/2023).
- [8] Mike Hanley. *We updated our RSA SSH host key*. <https://github.blog/2023-03-23-we-updated-our-rsa-ssh-host-key/> (visited on 05/16/2023).
- [9] Eyal Katz. *8 Proven Strategies To Protect Your Code From Data Leaks*. <https://spectralops.io/blog/8-proven-strategies-to-protect-your-code-from-data-leaks/> (visited on 02/01/2023).
- [10] Beata Berecki. *Best Practices for Source Code Security*. <https://www.endpointprotector.com/blog/your-ultimate-guide-to-source-code-protection/> (visited on 02/06/2023).
- [11] CheatSheets Series Team. *Secrets Management Cheat Sheet*. https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html (visited on 02/06/2023).
- [12] Mackenzie Jackson. *Best practices for managing and storing secrets including API keys and other credentials*. <https://blog.gitguardian.com/secrets-api-management/> (visited on 02/06/2023).
- [13] Kathy Charmaz. *Constructing Grounded Theory*. SAGE Publications, 2014.
- [14] Anselm Strauss and Juliet M Corbin. *Grounded theory in practice*. SAGE Publications, 1997.
- [15] Juliet Corbin and Anselm Strauss. “Grounded theory research: Procedures, canons and evaluative criteria”. In: *Zeitschrift für Soziologie* 19.6 (1990), pp. 418–427.
- [16] Melanie Birks and Jane Mills. *Grounded Theory: A Practical Guide*. Jan. 2015.
- [17] Cathy Urquhart. *Grounded Theory for Qualitative Research: A Practical Guide*. Jan. 2013.

Tables and Figures

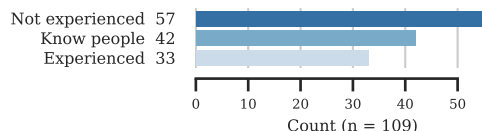


Figure 1: Developers reported experience on code secret leakage in the past or know people who did. We allowed multiple answers.

Table 1: Selected participant demographics from both the survey and interviews. We omit “Other” and “Prefer not to disclose” answers for space reasons.

	Survey			Interviews
	Upwork	GitHub	Combined	
Participants:				
Started	52	101	153	n/a
Finished	51	59	110	n/a
Valid/Total (n =)	50	59	109	14
Gender:				
Male	86.0%	88.1%	87.2%	92.9%
Female	10.0%	1.7%	5.5%	0.0%
Non-Binary	0.0%	6.8%	3.7%	7.1%
Age [years]:				
Median	29.0	33.0	30.0	28.0
Mean	31.3	34.9	33.2	32.1
Country of Residence:				
U.S.	2.0%	32.2%	18.3%	21.4%
India	20.0%	3.4%	11.0%	21.4%
Germany	0.0%	18.6%	10.1%	0.0%
Pakistan	14.0%	3.4%	8.3%	14.3%
Other	60.0%	40.7%	49.5%	42.9%
Development/Programming Education:¹				
Self-taught	92.0%	94.9%	93.6%	85.7%
College/University	54.0%	62.7%	58.7%	71.4%
On-the-job training	72.0%	42.4%	56.0%	57.1%
Online class	60.0%	28.8%	43.1%	35.7%
Coding camp	18.0%	8.5%	12.8%	0.0%

¹ Multiple answers allowed; may not sum to 100%.