Informatics

Security and Privacy Concerns in Shared Configuration Repositories

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Software Engineering und Internet Computing

eingereicht von

Gerhard Jungwirth, BSc
Matrikelnummer 01227311

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Assistant Prof. Dipl.-Ing. Dr.sc. Jürgen Cito
Mitwirkung: Assistant Prof. Dipl.-Ing. Dr.sc. Martina Lindorfer
Univ.Ass. Dipl.-Ing. Michael Schröder
Aakanksha Saha, BSc MSc

Wien, 4. Mai 2023

Gerhard Jungwirth   Jürgen Cito

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Security and Privacy Concerns in Shared Configuration Repositories

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Software Engineering and Internet Computing

by

Gerhard Jungwirth, BSc
Registration Number 01227311

to the Faculty of Informatics

at the TU Wien

Advisor: Assistant Prof. Dipl.-Ing. Dr.sc. Jürgen Cito
Assistance: Assistant Prof. Dipl.-Ing. Dr.sc. Martina Lindorfer
Univ.Ass. Dipl.-Ing. Michael Schröder
Aakanksha Saha, BSc MSc

Vienna, 4th May, 2023

Gerhard Jungwirth   Jürgen Cito

Erklärung zur Verfassung der Arbeit

Gerhard Jungwirth, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 4. Mai 2023

Gerhard Jungwirth

Danksagung

Mein Dank geht an meine Betreuer:innen, Martina Lindorfer und Jürgen Cito, die dieses Projekt ermöglicht haben und mir die Chance gaben, in diesem Team zu arbeiten. Weiters möchte ich mich bei den anderen Teammitgliedern, Michael Schröder und Akanksha Saha, für ihr Feedback, Hilfestellungen und Tipps bedanken. Außerdem danke ich Katharina Krombholz für ihre Beratung betreffend die empirische Umfrage und Auswertung. Zum Schluss möchte ich mich bei meiner Familie und allen Freund:innen bedanken für all die Motivation und Unterstützung, die sie mir gegeben haben. Besonderer Dank geht hier an Laura, Sandra, Lisa, Michi, Peter und Marcel für ihre Freundschaft und immerwährende Unterstützung in allen Belangen.

Acknowledgements

First of all, I want to thank my advisors, Martina Lindorfer and Jürgen Cito, for making this project possible and giving me the opportunity to work on this subject. I also want to thank the other members of our team, Michael Schröder and Akanksha Saha for their help, tips, and feedback. In addition, I want to thank Katharina Krombholz for her counseling regarding the empirical survey. Finally, I want to thank all my friends and family for their endless hours of support and motivation. Special thanks go to Laura, Sandra, Lisa, Michi, Peter, and Marcel for their friendship and incredible support over the last years.

Kurzfassung

Der Begriff „dotfiles“ bezeichnet Textdateien, die zur Konfiguration in UNIX Systemen genutzt werden. In den letzten Jahren hat sich eine Gemeinschaft gebildet, in welcher sich über solche Softwarekonfigurationen ausgetauscht wird. Repositories mit solchen Dateien sind auch auf GitHub zu finden und können als spezifische Gruppe betrachtet werden. Obwohl diese Dateien sehr personalisiert sind, enthalten sie auf den ersten Blick keine sicherheitskritischen Informationen. Bei näherer Betrachtung ergeben sich aber eine Reihe von möglichen Problemen die zu beachten sind: von Passwörtern und API-Schlüsseln hin zu persönlich identifizierbaren Daten (PII). Wir verwendeten zwei verschiedene Ansätze, um dieses Phänomen zu quantifizieren. Um das Ausmaß und die Art möglicher Sicherheits- und Privatspähreprobleme zu erörtern, führten wir eine Analyse von 124 230 öffentlichen Repositories auf GitHub durch. In einem iterativen Prozess fanden wir mögliche Lücken in 73,6 % der Repositories. In einem zweiten Schritt kontaktierten wir die betroffenen Author:innen und führten zeitgleich eine Umfrage (n=1 650) durch, um die Motivation für das Teilen von Konfigurationsdateien und das Wissen bezüglich Sicherheit und Privatsphäre zu erörtern. Wir fanden heraus, dass wichtige Gründe für das Teilen ideologischer Natur sind, eine soziale Ursache haben (zum Beispiel Angeben) oder in Bequemlichkeit zu finden sind. Die meisten Benutzer:innen sind zuversichtlich, was die Sicherheit ihrer Daten betrifft und geben an, die Sicherheitsaspekte gut zu verstehen. Schlussendlich geben wir noch Empfehlungen an Plattformbetreiber:innen und Nutzer:innen um dieses Feld sicherer zu gestalten.

Abstract

Dotfiles is a common term for text-only configuration files in the UNIX ecosystem. A community of sharing and exchanging of these files has formed in the last years. Configuration files, therefore, constitute a highly idiosyncratic domain of GitHub repositories. They are highly personalized but do not pose a potential threat at first glance. But on closer examination, they may provide a range of potential pitfalls, from passwords and API keys to personally identifiable information (PII). We used a twofold approach to analyze these issues. To find out the extent and existence of such issues, we conducted a large-scale analysis of 124,230 public dotfiles repositories on GitHub. In an iterative process, we found issues in 73.6 % of the repositories. We then contacted the authors and did a large-scale survey (n=1,650) on their motivation for sharing and their security considerations. We found that the main reasons for sharing are ideological, to show off ("ricing"), and to ease machine setup. Most users are confident about the contents of their files and claim to understand the security implications. After our search, we provide recommendations for Platforms as well as users to make this domain more secure.

Contents

CHAPTER 1

Introduction

GitHub is an online service and host for collaborative software development. It is widely used in open source software development, reaching a reported 100 million live repositories in November 2018 [49]. Apart from allowing users to publicly share versioned content of any type, it also includes collaborative and social functionalities. This has recently led to the trend, that users share their personal software configurations on GitHub. These repositories are typically called “dotfiles”. This name is due to the fact, that in UNIX-based systems, software configuration files usually start with a dot. In this way, they are hidden in the default file view. A detailed explanation of dotfiles and this trend can be found in Section 2.1.

Software engineering research was performed by extracting and mining large-scale datasets from GitHub and other source code hosts. A wide variety of qualitative and quantitative studies have been conducted to assess social interaction, development practices, and processes. Furthermore, security research has been conducted on this data. This has, however, primarily focused on software development projects. But many repositories on GitHub are not primarily part of software development projects [17].

Our research tries to close that gap and analyze possible security-relevant aspects of shared configuration repositories. These repositories contain information, that is closely linked to individual users, therefore the potential for vulnerabilities is high. Other research has shown that sensitive data is committed to GitHub regularly by accident to dotfiles as well as other repositories [23].

An example of the effects of security leaks in public repositories is the "2020 United States federal government data breach". Besides US government departments, the European Union, NATO, Microsoft, and others were affected. While numerous attack vectors were used, one that could have played a role is a vulnerability in Orion, a popular network monitoring software of SolarWinds. As a security researcher pointed out, the credentials to their update and build system have been leaked on GitHub already a year earlier [18, 38, 46].

In this paper, we investigate the risks of publicly sharing personal configurations on platforms like GitHub. In particular, we want to answer the following research questions:

RQ1 What are the security and privacy implications of public shared configuration repositories (a.k.a. dotfiles) on platforms like GitHub?

RQ2 What are the primary motivations of repository owners for sharing their dotfiles?

RQ3 How aware are repository owners of the implications of sharing their dotfiles?

To this end, the contributions of our work are:

- **A large-scale analysis of 124,230 public dotfiles repositories** on GitHub, revealing 10,942,606 potential security and privacy threats, affecting 73.6 % of analysed repositories. These include leaked email addresses, RSA private keys, API keys, history and log files, and more. We classified the types of leaks and the possible attack vectors, such as hijacking, impersonation, spamming, or phishing.
- **The responsible disclosure to 5,199 repository owners** of relevant issues, wherever this was possible.
- **A survey of 1,650 repository owners** to understand their motivations and their perceptions of security risk. Our results show a common intent of *sharing and learning* from each other, with an added motivation of showing off one's highly configured system configuration ("ricing"), as well as *convenience* when synchronizing and restoring dotfiles. Our findings indicate that users readily understand the risk of sharing sensitive information through dotfiles, and a majority of participants will continue sharing their dotfiles. Our survey materials and questionnaires are available in our appendix.

Chapters 2 and 3 cover the background research and related work. Next, Chapter 4 describes the process and results of the in-depth iterative analysis of dotfiles-repositories on GitHub, while Chapter 5 explains the survey on dotfile-repository owners on GitHub. Finally, we close with a conclusion and proposals for further work.

CHAPTER 2

Background

2.1 UNIX Configuration Files

UNIX-based systems usually follow a number of standards. One of them is, that files that start with a dot are hidden by default when viewing the contents of a directory. It, therefore, has become a common practice for tools, to store user-specific configurations in such a file in the user's home directory (e.g `.profile` for shell configurations or `.emacs` for configurations of the EMACS file editor). Due to these properties, they are typically called "dotfiles" [51].

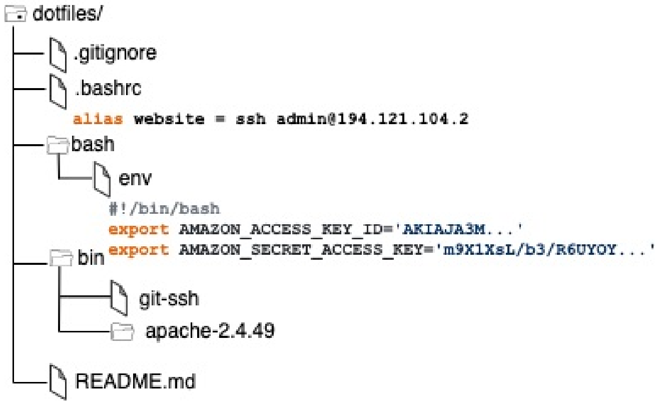
Originally, these files were simple text files, that were written by the user and read once during program initialization. With the emergence of complex graphical desktop environments and "modern" GUI applications, hidden files, and folders started to contain a lot of stuff (e.g. extensive default configurations, runtime and cached data, user/session data, named pipes). It, therefore, became harder to synchronize these configurations between different machines and software versions [48].

Nonetheless, users have started to use version control systems and dedicated tools to manage these files which became increasingly complex. An overview of these approaches will be shown in the next section.

2.2 Distributed VCS and Trends

Version control systems are a popular technology in software development and other IT-related applications for fine-grained tracking of changes and authorship – especially in plain-text files like source code. One of the most popular choices is Git[1], originally developed by Linus Torvalds for the Linux Kernel. One of the most popular online

[1] `https://git-scm.com/`



```
alias website = ssh admin@194.121.104.2
```

```
#!/bin/bash
export AMAZON_ACCESS_KEY_ID='AKIAJA3M...'
export AMAZON_SECRET_ACCESS_KEY='m9X1XsL/b3/R6UYOY...'
```

Figure 2.1: Structure of a typical dotfiles repository exposing sensitive information that brought together could form an attack vector

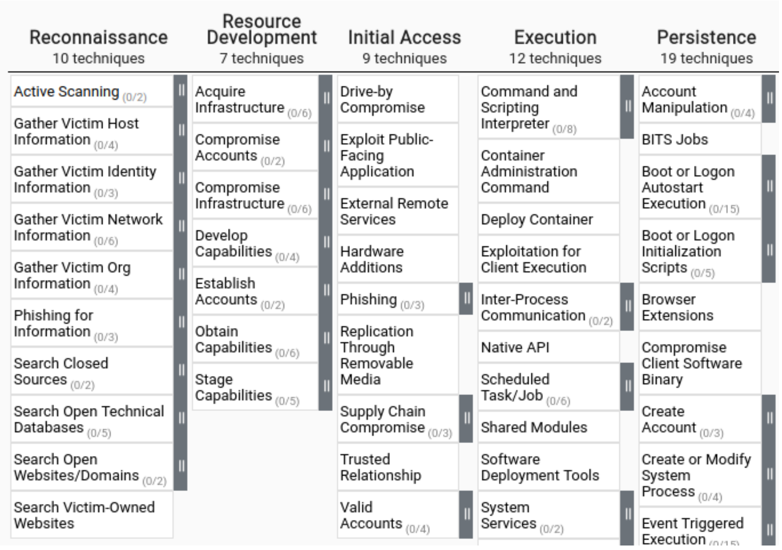
platforms to host Git repositories is GitHub. It allows private users free online storage for public and private repositories.

In order to share their personal configuration files with others or synchronize them onto different machines, users have recently started to share their dotfiles on platforms like GitHub. These are typically personal repositories which are often named "dotfiles". An example of such a dotfiles repository structure can be seen in Figure 2.1. A search for the string `dotfiles` shows 162,964 results in November 2021. A community has evolved out of this trend, promoting a sense of sharing and collaboration. Dotfiles are often copied and cloned from popular users and a number of tools have been developed to manage them [26]. Groups on Reddit (like the board `r/unixporn`) also often share their dotfiles in an effort to establish subculture identity and credibility.

The Reddit board `r/unixporn` also coined a term called ricing - which means showing off a good, often self-made desktop configuration [12].

2.3 Security Basics

The field of scientific security research often concentrates on two areas: Formal Methods to prove the absence of certain risks or an applied approach of finding new attack surfaces, privacy problems, or other characteristics of a system, protocol, or technology or evaluate these properties therein. In order to categorize vulnerabilities, threats and attacks the Open Web Application Security Project (OWASP) publishes various lists of the most frequent issues in different fields. Another categorization is provided by the MITRE corporation – the ATT&CK framework [43]. It lists tactics, techniques, and documented usage of those and may be used to classify actual attacks in retrospect. Figure 2.2 show an exemplary part of the MITRE classification tree.



Reconnaissance 10 techniques	Resource Development 7 techniques	Initial Access 9 techniques	Execution 12 techniques	Persistence 19 techniques
Active Scanning (0/2)	Acquire Infrastructure (0/6)	Drive-by Compromise	Command and Scripting Interpreter (0/8)	Account Manipulation (0/4)
Gather Victim Host Information (0/4)	Compromise Accounts (0/2)	Exploit Public-Facing Application	Container Administration Command	BITS Jobs
Gather Victim Identity Information (0/3)	Compromise Infrastructure (0/6)	External Remote Services	Deploy Container	Boot or Logon Autostart Execution (0/15)
Gather Victim Network Information (0/6)	Develop Capabilities (0/4)	Hardware Additions	Exploitation for Client Execution	Boot or Logon Initialization Scripts (0/5)
Gather Victim Org Information (0/4)	Establish Accounts (0/2)	Phishing (0/3)	Inter-Process Communication (0/2)	Browser Extensions
Phishing for Information (0/3)	Obtain Capabilities (0/6)	Replication Through Removable Media	Native API	Compromise Client Software Binary
Search Closed Sources (0/2)	Stage Capabilities (0/5)	Supply Chain Compromise (0/3)	Scheduled Task/Job (0/6)	Create Account (0/3)
Search Open Technical Databases (0/5)		Trusted Relationship	Shared Modules	Create or Modify System Process (0/4)
Search Open Websites/Domains (0/2)		Valid Accounts (0/4)	Software Deployment Tools	Event Triggered Execution (0/15)
Search Victim-Owned Websites			System Services (0/2)	

Figure 2.2: Exemplary section of the MITRE ATT&CK framework classification [24]

CHAPTER 3

Related Work

3.1 Mining GitHub

Kalliamvakou et al. [17] present an overview of the benefits and pitfalls when mining GitHub. They showed that repositories are not only used for active software development; rather, users have personal repositories for different purposes, many of which are inactive. According to their survey, other use cases include experimentation, website hosting, school and university projects, as well as just storage.

Since GitHub offers many social features, and collaboration is a major use case of distributed version control, a number of studies have targeted social interaction and collaboration on GitHub, focusing on the frequency, type, and duration of collaborators' participation: Tsay, Dabbish, and Herbsleb [44] measured the likelihood of acceptance of pull requests on open-source projects on GitHub based on social and technical parameters. They show that social ties and prior interaction of the involved developers with the project strongly contribute to an accepted pull request.

Casalnuovo et al. [9] took a look at the onboarding process of new project members. They found out, that prior social ties as well as language experience are two factors that improve the chances of prolonged participation.

Pinto, Steinmacher, and Gerosa [29] also followed a mixed-methods approach including two surveys (n=197 and n=64). They intended to examine the work of casual contributors to open-source projects including their motivation, their type of contributions and their share of the overall contributions. They conclude that casual contributors are an important part of many projects and they deliver many non-trivial fixes and improvements. A main motivation for casual contributors is a personal need.

Qiu et al. [32] conducted a mixed-methods empirical study to find out, whether social capital has a different effect on the dropout rates of male and female developers at open-source projects. They found out, that diversity across various categories is beneficial to prolonged participation and has a positive effect on gender differences as well. The survey received n=88 responses which were evaluated.

Other papers have questioned how people navigate and evaluate projects and how this impacts their inclination towards contribution [7, 54, 22]. Ray et al. [33] have analyzed the relationship between code quality and programming language choice, while Schröder and Cito [39] investigated the use of shell aliases.

Our research enriches these endeavors in that it mines GitHub repositories for a specific subdomain going further into detail on the possible ramifications for that domain (being configurations repositories).

3.2 GitHub Security Research

A number of studies have focused on source code vulnerabilities and security issues surrounding the software development process [25, 55, 13]. Lazarine et al. [20] leverage graph embedding algorithms to study networks and relationships of vulnerabilities in GitHub repositories to better identify vulnerable projects and organizations. Closer to our interests, Yasar [52] investigated secrets that are accidentally committed into repositories as part of continuous integration (CI) pipelines, and Meli, McNiece, and Reaves [23] have performed the first large-scale analysis of secret leakage in GitHub repositories. They found that a significant amount of API keys and private key files in existing repositories on GitHub, and that thousands of new leaks are committed every day. They also found that a purely regular-expression-based search delivers many false positives, however—mostly test keys and placeholder strings. To validate their findings, they used a combination of filters based on entropy and pattern detection. Saha et al. [36] and Lounici et al. [21] use machine learning to improve detection accuracy and eliminate more false positives.

These references show that the existence of secrets in source-code repositories has been well-researched and multiple approaches have been taken to improve the matching accuracy. In comparison, our work explores further attack vectors and provides an overview of possible attacks using a combination of those individual threats. We also show that the domain of personal configuration files has some differences from plain source code repositories as it may contain more personal data and individualized content.

3.3 User Surveys Related to Security and Privacy

Several surveys have targeted users' perceptions and knowledge of security and privacy in the field of online software repositories and distributed software development.

Bühlmann and Ghafari [8] conducted a long-term, large-scale analysis of discussions on security reports between 2014 and 2020. They studied security issue reports on 182 repositories and found that the rate of new reports was increasing over time. On the other hand, there is a large number of reports which are not resolved, or the time until they are resolved is rather long. Issues are rather reported and handled by a small fraction of core “expert” developers.

Dietrich et al. [11] performed a survey where they investigated the system operators’ perspective on security misconfigurations. They did this via a qualitative and a quantitative survey. One finding was, that two third of security-relevant misconfigurations exist for a long time, without ever leading to an actual incident, although they were critical. They also propose some measures to prevent such configuration mistakes and lead to more secure systems.

Pletea, Vasilescu, and Serebrenik [30] analyzed the sentiment of participants in security-related discussions on the platform GitHub. They found that about 10% of all discussions on GitHub are on a security-related topic. These discussions typically contain more negative emotions than other discussions. As a result of their findings, they propose to address security throughout the whole project lifetime, take measures in improving discussion culture and handle upcoming issues professionally in order to improve the general atmosphere and increase overall security.

Alqhatani and Lipford [2] used semi-structured interviews with 30 users of wearable fitness devices in order to explore their practices of sharing the data generated on those devices on different (social) platforms. They found that most of the users are more concerned with self-presentation and social norms than about the sensitivity of the data itself.

Kariryaa et al. [19] conducted an online survey (n=353) to evaluate users’ understanding of the security aspects of browser extensions, in particular the reception of permission dialogues. Their conclusion was that users have little knowledge of the actual capabilities of browser extensions and that current designs are insufficient in helping users understand the impact of their choices.

Wermke et al. [50] conducted surveys with 200 users of Cloud Office suites to investigate their understanding, perception, and expectation of the security and privacy of those products. While users seemed to have strong opinions on how their documents should be shared and to whom they should be visible, they often do not know whether this expectation is met in reality. There was a lack of technical understanding in some cases but on the other hand, a general awareness of the implications of cloud solutions was available. Wermke et al. [50] also provide some recommendations to the different parties involved so that cloud office users can make an informed decision about their security and privacy.

Ponticello, Fassl, and Krombholz [31] conducted semi-structured interviews (n=16) in order to evaluate users’ risk-assessment when performing sensitive tasks like home banking on voice-controlled assistants. They found that users are less willing to use token-based authentication, e.g. via a smartphone, as it undermines the convenience of smart-home appliances. Users strongly favor biometric authentication methods, although some are aware of their flaws in current systems.

Bailey, Markert, and Aviv [3] conducted an online survey (n=235) to find out users' perceptions and understanding of the PIN feature of the messaging app Signal. They found that only enthusiasts have a full understanding of the PIN as an account recovery key and choose a sufficiently strong alphanumeric one, while casual users do not understand the purpose of this feature and just set a minimal 4-digit numerical code. They conclude that phrasing and in-app explanations are crucial for such a security-relevant feature. Vasilescu, Filkov, and Serebrenik [47] conducted an online survey (n=816) in order to understand collaboration factors on GitHub participation. They found that GitHub with all its social features and transparent contribution options (like pull requests) provides a low barrier for newcomers and that teams largely benefit from diversity across technical and social categories in their group of contributors.

CHAPTER 4

Quantitative Research

In this chapter, the first part of our research efforts is presented: the exhaustive mining of GitHub for dotfiles-repositories. First, the goals and methodology are presented. Then the results are described and finally, we present a discussion of possible reasons and implications.

4.1 Research Goals

As we described in the previous chapters, there has not been a study so far that specifically addressed the security and privacy concerns of shared configurations. Therefore, our goal was to perform an iterative, exhaustive search of dotfiles repositories on GitHub and evaluate their contents with a focus on security and privacy issues. A special interest lies in the question, of whether there are also more complex attack vectors apart from one-dimensional ones like leaked credentials.

4.2 Methodology

Here we describe how we defined the target boundaries and how we collected the resulting data. After that, we will explain the general approach of processing this data. In general, we conducted our research on 124,230 repositories, which have a total size of 993 GB. These include the full history, as well as a checked-out working directory for each repository. They have been downloaded between October 2020 and January 2021.

In the next two sections, we will describe this process in more detail.

4.2.1 Data Collection

Our goal was to search and analyze shared configuration repositories, also known as “dotfiles”. See section 2.1 for a description of these files and repositories. We found that not everyone who shares a repository containing dotfiles also gives it the name “dotfiles”. Therefore we used the GitHub repository search API [16]. In contrast to the code search API, this endpoint delivers reliable, complete information. We, therefore, queried the API to search for the string “dotfiles” in either the name or description of the repository.

Nonetheless, the API has two more limitations. First, a number of (variable) rate limits apply, to prevent abuse and secondly, each query only delivers a maximum of 1,000 results. We therefore facilitated a custom-made API tool, developed at TU Wien which regularly queries the rate limits and dynamically set delays to maximize throughput and modified it to use the repository API[1,2]. To circumvent the limit of 1,000 results per query the results are split by adding search parameters for the size of the repository and repeating the query while varying this parameter to reach a result set below this limit per call. Also, we excluded forks from our search to limit the result space and get a broad insight into the domain without dealing with duplicates.

We began our repository enlisting on October 10th, 2020, and took about ten hours including some manual adjustments to the search parameters. Querying the GitHub API with these properties – “dotfiles” in name or description, excluding forks – in an exhaustive search resulted in 125,171 repositories, that matched our criteria. Out of these, the majority (113,860, about 91 %) included the string “dotfiles” in the repository name.

We then used `git clone` with ssh public key authentication and a small delay to download these repositories. Due to technical constraints on the server used for our experiments, we ran our cloning process in two batches. We started the first cloning process on October 28th, 2020, and collected 75,710 repositories. We started the second batch on January 4th, 2021, and collected a remaining 48,520 repositories. Out of 125,171 repositories, 884 found via the API search tool could not be cloned, probably because they had been deleted in the meantime. We further excluded 56 repositories from our analysis because of their size (greater than 1 GB).

About 46 % of the total repositories we cloned had their last commit in 2020, and we therefore consider them active. On average, each repository has 81 commits and three different authors — although the number of authors per repository can be misleading, as the same person can commit to a repository using different authorship information. This has also been shown by other research [14] and was confirmed by us through manual inspection.

4.2.2 Data Processing

After the cloning step, we collected some general metadata and stored it in an SQLite database. We will present the dataset below and give an overview of how it looks like and what it contains. All data was stored in an access-controlled server at university

[1] https://github.com/ipa-lab/github-searcher

[2] https://github.com/ipa-lab/dotfiles-repositories-analysis

Table 4.1: Frequency of MIME Types (Top10)

MIME Type	# Files	% Files	Avg. Size	Avg. Lines
text/plain	9,235,370	46.29 %	9.8 KB	193
image/svg+xml	3,820,771	19.15 %	3.4 KB	38
image/png	1,955,465	9.80 %	23.5 KB	85
application/octet-stream	635,244	3.18 %	82.0 KB	464
text/x-shellscript	580,310	2.91 %	2.8 KB	63
application/json	507,723	2.54 %	7.6 KB	116
text/html	488,127	2.45 %	10.5 KB	178
text/x-lisp	306,868	1.54 %	17.3 KB	436
text/xml	294,727	1.48 %	25.8 KB	501
text/x-python	208,948	1.05 %	9.1 KB	247

Table 4.2: Frequency of File Names (Top10)

filename	# Files	# Repos
`README.md`	333,164	69,845
`.gitignore`	85,550	51,734
`.vimrc`	42,180	41,180
`.zshrc`	34,117	33,219
`.gitconfig`	29,071	28,587
`config`	59,760	28,554
`.tmux.conf`	26,565	26,061
`.bashrc`	26,970	25,981
`vimrc`	23,718	22,672
`.gitmodules`	20,123	19,018

premises. To get an overview over the data, we collected some statistics. We indexed all files that were present at the latest state of the repositories (working copy). For these files we analyzed their size and number of lines (which is only useful in the case of text files). We also used the UNIX file utility[28]. which uses a heuristic to determine the MIME type and stored the file type for each file in all repositories. In total, we analyzed about 20 million files. We found that the majority of files in these repositories (61 %) were textfiles like text/plain, text/x-shellscript, application/json, and so on. A more detailed overview of the ten most common MIME types can be found in Table 4.1. The most common file name was `README.md`, appearing in more than half of the repositories, followed by `.gitignore` and typical editor and shell configuration files like `.vimrc` and `.zshrc` (see Table 4.2).

We also tried to index the contents of all text files to identify typical or common content. We used a fuzzy hashing algorithm to also match similar files. For this purpose, we intended to use ssdeep[3], a very common algorithm/tool, which is among others used in malware detection. Nonetheless, the size of our data made it impossible to complete the matching. Therefore, this process was not completed and hence can be explored in the future.

4.2.3 Data Analysis

After collecting and processing the dotfiles repositories, we identified potential security and privacy threats (**RQ1**). Related work has already focused on identifying authentication credentials in repositories [23, 36, 21]. We extend this line of work, in particular building on Gitleaks [34], which searches a repository and its history based on regular expressions. We sampled our dataset and iteratively identify further relevant types of information that are common in dotfiles. We crafted regular expressions for each of them and integrated them into Gitleaks to identify them in our dataset at scale.

We further map the identified security and privacy issues against the MITRE AT&CK framework [24]. This knowledge-base documents attacker tactics and techniques and is commonly used in industry to understand attacker models and methodologies. In particular, we discuss how the individual pieces of information available in the repositories could aid attackers in the reconnaissance phase of their attacks.

4.3 Results

We now describe the security and privacy issues we found in our dataset. An overview of the found quantities can be found in Table 4.3.

4.3.1 Generic Credentials

A well known problem of GitHub and similar platforms is, that authentication credentials are accidentally committed into repositories [23, 36, 21]. This can be in the form of username/password, API tokens or asymmetric private keys. We used the tool Gitleaks[34]. to find such occurrences. In its default configuration, it uses a number of regular expressions to find common API keys of popular services such as the Google APIs as well as private keys. An advantage of this tool over a plain fulltext search is that it also searches the full history of each repository.

We identified possible leaks in 11,758 repositories (which corresponds to 9.5 % of all repositories). The most common credential was a GitHub API key, followed by Twitter. Table 4.4 gives an overview of all the credentials found in our scan.

To some extent, these results represent the general popularity of the different platforms, and are of course constrained by the regular expression set we used. Similar findings have been reported by Meli, McNiece, and Reaves [23], although they found Google API keys to be the most commonly leaked credentials, which also matches the GitGuardian

[3] `https://ssdeep-project.github.io/ssdeep/index.html`

Table 4.3: Overview of all findings after mining dotfiles repositories on GitHub. We quantify the prevalence of particular security and privacy relevant information in these repositories. We also note where some of our findings replicate existing studies and which possible attacks are represented by these findings. The number of vulnerabilities is counted once per file.

Type of Information	#	Repos (%)	Notes	Possible Attacks
API Keys				Hijacking, Impersonation, Spamming
Github API keys	65,589	6,898 (5.51 %)	Example: ghp_16C7e42F292c6912E7710c838347Ae178B4a[4]	
Twitter Key	38,752	3,936 (3.14 %)	[23]: 20,760 keys	
Other	19,166	4,470 (3.57 %)		
RSA Keys				Hijacking, Spamming
RSA Private Key	9,452	1,489 (1.19 %)	[23]: 158,011 keys	
Public Weak Key	111	n.a.	Key length $\leq$ 1024 bit	
Public Vulnerable Key	6	n.a.	Debian RNG attack [53]	
Software Packages				Hijacking
Python Dependencies	16,315	1,036 (0.83 %)		
Javascript Dependencies	145,050	585 (0.47 %)		
Private Data				Hijacking, Impersonation
Firefox Logins	40	29 (0.02 %)		
Thunderbird Profiles	2	2 (0.002 %)	Actual user data, not metadata	
Mailboxes	52	52 (0.04 %)	Inbox files from Thunderbird and Mutt	
PII				Spamming, Phishing
Email Addresses	1,227,175	88,442 (70.7 %)		

2021 report [15]. The difference probably stems from the fact that our research focuses on personal configuration repositories, which are typically used for non-code purposes.

These matches definitely include false positives. This became evident by reviewing the file paths or the matching line of the results for components like "dummy" or "test". For example, this string was found in the filename a total of 6,294 times or 4.73 % of all matches. We did not further focus on differentiating true and false positives, since this has already been done by other research with various means, including machine learning [36, 21]. On the other hand, a significant amount had no such indicator and must therefore be counted as real credential.

4.3.2 Weak Public Keys

Apart from 9,452 RSA private keys, which obviously should never be shared, we also found a number of problematic public keys. Normally, public keys are supposed to be shared [4] and GitHub itself even provides access to all public keys of its users. Nonetheless, if used incorrectly, asymmetric encryption can be weakened and poses a potential threat [53].

We analyzed two kinds of files: Files with the string `id_rsa.pub` and files with the

Table 4.4: Number of matched Gitleaks Rules

Count	Rule
65,589	Github
38,752	Twitter Client ID
9,452	Asymmetric Private Key
4,981	LinkedIn Client ID
2,880	Google API key
2,761	AWS Access Key
2,169	AWS Secret Key
2,051	LinkedIn Secret Key
1,557	Facebook Client ID
1,255	Twitter Secret Key
657	Facebook Secret Key
560	Slack
149	Google (GCP) Service Account
85	Slack Webhook
39	Mailgun API key
11	SendGrid API Key
7	MailChimp API key
3	Stripe API key
1	Picatic API key

string `authorized_keys` in their file name or path. In total we found 1,566 repositories where one of those conditions were true. Files with the first described format are public key files belonging to a key pair of a user, typically found in their `.ssh` folder. We already analyzed these folders when analyzing the private keys and found no further interesting properties in these files. The second type of file also contains public ssh keys and is typically found in ssh server configurations. The public keys which are listed in that file, grant the owner of the respective private key access to that server, depending on the configuration usually without a second factor. We found 1,050 such files. In those, we found 2,844 parseable RSA public keys, 192 DSA keys and 109 ECDSA keys. Out of the RSA keys, 111 had a key length of $\leq$ 1024 bit. This is a potential security risk, depending on the criticality of systems, which are accessed by it [5]. We also verified, if any of these keys are vulnerable to an attack, where an insecure random number generation had led to weak keys in the past [53], by comparing them to an available dataset of affected keys[5]. In this process, we found six affected keys. This is a relatively low number and shows that the vulnerability was first discovered 2008 and has had a lot of publicity over the years. After a discovery in 2015 that a lot of users still use those weak keys [10], GitHub has proactively started to check the keys, which were used to log in to the platform and invalidate the weak ones they found.

[5] `https://github.com/g0tmi1k/debian-ssh`

Table 4.5: Cryptographic algorithm used in SSH public keys

#	Algorithm
2,844	RSA
192	DSA
109	ECDSA

4.3.3 Personally Identifiable Information (PII)

An obvious issue with personal repositories is that they may contain a lot of personally identifiable information (PII). We found that we could extract, among others, the following information out of the dotfiles dataset:

- Personally Identifiable Information
  - Name/Email/Username
  - Language
  - Geographical Information
  - Used OS and Software
- Organizational Information
  - Workplace or Place of Education
  - Internal Networking Infrastructure

This information could be mined at a large scale and could be used as intel for different attacks like spear phishing or impersonation attacks or sold for other purposes.

As an example, we extracted all email addresses from the repositories, which are a huge source of information themselves, e.g.: usernames, social graph, company affiliations, country. In order to extract all emails from the timelines of all repositories, we configured Gitleaks to use our own regular expression to extract email addresses. In total we extracted 22 million email addresses from our dataset. If every unique email address is counted only once per file, this still gives us 9.3 million email addresses. Table 4.6 shows the most common email domains. In order to evaluate the significance of each email address, we calculated the TF-IDF metric [35]. This gave us a hint on the significance of each email and highlighted the most important addresses for each repository. We found 1.2 million unique non-trivial email addresses in total.

In the course of this analysis, we found that in some cases people have included whole email inboxes into their repository (i.e., files with received and sent raw email messages as they are typically stored by email clients). This was most likely unintentional. We found 23 repositories, where this was the case. After an exhaustive search with email

Table 4.6: Top five most common email domains

#	Domain
824,753	gmail.com
35,080	github.com
28,224	example.com
24,046	yahoo.com
20,351	gnu.org

Table 4.7: Frequency of occurrence of certain software profiles

Software	#repos
Firefox	361
Google Chrome	152
Discord	46
Skype	12

inboxes in mind, we discovered 52 files with valid emails in total. Most of these inboxes were found in historic states of the repositories and have been deleted. Nonetheless, they were still present in the git history. We will discuss the issue of sensitive historic data in section 4.4.5.

4.3.4 Private Data

During the iterative refinement of our qualitative analysis, it became evident, that some of the repositories also contain unintentional user data. Apart from the email inboxes described in section 4.3.3, we also found private data from web browsers and chat programs. This data includes, among others, cache data, cookies, browsing history and even unencrypted passwords from the browsers' password managers. We developed queries to systematically search all repositories for the frequency of occurrence of such data and found that between 46 to 361 repositories contain such data – depending on the software that we searched for. This corresponds to a maximum of 0.3% of all repositories but we still consider it a significant problem due to the severity of such cases. Table 4.7 shows an overview of the prevalence of these findings. Through manual inspection we found that our data contains false positives (where no sensitive data is present and the profiles contain only plain configuration) as well as true positives.

We also saw that our data contains IRC (internet relay chat) logs. Since these chat rooms are often public and even logs are often publicly available, it was not possible to determine the privacy status of this data.

4.3.5 Software Packages

By analyzing common software dependency files, we can infer that particular users are using certain software packages. Several attack vectors are associated with this kind of information. Typosquatting and dependency confusion have recently gained the attention of security researchers. Both can be rated as supply chain attacks [27]. Typosquatting has been popular with domain names for over two decades, where similar domains to an existing one (like “netflix.om” instead of “netflix.com”) have been used to spread malware or other malicious activities [40]. With package dependencies, it is even easier because anyone can typically upload packages to package repositories like PyPI and it has been proven to be a very effective attack [45]. Dependency confusion is a new technique that is supported by the fact that big companies often have a mix of open-source and private packages which they use internally and which are installed by common package managers from a mix of public and private package archives. The problem is that the package managers often prefer the highest version of a package so malicious packages with the same name as an internal dependency of a company can be uploaded to public archives and then have the same impact as with typosquatting – or even a higher impact since this is typically used by larger companies [6].

Therefore – even though they are probably not as relevant to shared configuration repositories as they are for active software development projects – we wanted to analyze library dependencies. We found that the main source of structured dependency information comes from Python and JavaScript dependency definitions. Even though only a maximum of about 2% of all dotfiles repositories contain such dependency files, vulnerabilities can still have a high potential impact.

We cross-checked all found dependencies with the publicly available package archives of the respective languages to find out, how many packages are not listed there (they may be installed from another source or be simple typos). The majority of those packages just came from other sources than the official package indices and may not immediately pose a vulnerability. They still may be vulnerable, depending on how the installation process is done. In order to find actual typos, the Levenshtein distance was used to find similar packages for those, which could not be found in the archive. It was possible to identify clear typos in this way, which would be susceptible to typosquatting.

We also analyzed how dependency versions were specified on the example of javascript dependencies. The majority of version definitions were specified with the caret `^version` (62%), which only increments the patch version of that package. 14 percent used the tilde (`~version`) or an exact version specifier (`version`) respectively. Especially the latter may be problematic because important security fixes could be missed.

Table 4.8: MITRE AT&CK classification of our findings [24]

TA0009: Collection	T1602: Data from Configuration Repository	
TA0043: Reconnaissance	T1592: Gather Victim Host Information	
	T1589: Gather Victim Identity Information	
	T1590: Gather Victim Network Information	
	T1591: Gather Victim Org Information	
	T1593: Search Open Websites/Domains	
TA0006: Credential Access	T1555: Credentials from Password Stores	.003: Credentials from Web Browsers
	T1552: Unsecured Credentials	.001: Credentials in Files
		.004: Private Keys

4.4 Discussion

4.4.1 Security Implications

Our research has shown, that dotfiles are a sensitive example of public data on GitHub. Even though they are supposed to contain personal data, they are typically shared with the public intentionally. The existence of (API) keys and credentials is documented [23, 36, 21] and there exist several solutions including GitHub's Code Scanning[6], yet secrets are still present at large scale in users' repositories. Besides secrets, we found a range of other information with a potential of exploitation by a malicious actor, especially during reconnaissance, which is an important phase of contemporary attacks [1]. This information can be seen as host information (like credentials), identity information (like emails), network information or organisational information. Table 4.8 maps our findings to the relevant attack stages and classes in the MITRE AT&CK framework [24]. Private data (browsing history, emails...) and personally identifiable information are types of data that have privacy implications and can indirectly even be used in the reconnaissance for further targeted attacks (e.g., by feeding brute force password lists). On the other hand, we also found data that leads to more direct attack surfaces: credentials, vulnerable dependencies, and weak keys. These can be used directly for targeted attacks. The combination of direct and indirect data can lead to targeted and sophisticated attacks as the attacker has better familiarity with the target's infrastructure. For example, knowledge of a vulnerable web server package coupled with a domain credential can lead to privilege escalation within a target environment (see Figure 2.1). In the next section, we are going to lay out a few attack scenarios that pose a high threat.

4.4.2 Possible Attacks

The highest potential threat lies in a combination of the data that we found and described in our findings. The reason for that is, that dotfiles repositories naturally contain highly personal information which provide potential for attacks individually and even more so in combination. Here we want to lay out some examples of what might be possible attack strategies with the available data. The availability and occurrence of combinations of

[6] https://docs.github.com/en/code-security/secret-security/about-secret-scanning

the different features we analyzed is visualized in Figure 4.1. For example the middle number means that five repositories contain at least one item of all five categories of findings that we analyzed. The five categories correspond to the findings we described in Section 4.3. This graphic just serves as an overview and proof, that the “features” we found can be combined in various ways.

Summarizing our findings, dotfiles repositories can – among others – enable the following potential attack scenarios:

- *Credential Stuffing:* Malicious actors can simply use the API credentials to get access to the respective services. Alternatively, they can use the found passwords or private keys (and possibly usernames, emails, and other data) in combination with host information to authenticate against the target services (e.g., cloud computing and storage services).
- *Vulnerable Packages:* Vulnerable packages can be identified from the package information, combined with host information can consequently give access to a specific system. For example, an attacker identifies a web server vulnerability that allows for unauthenticated access to data stored within a web server.
- *Impersonation:* Personal, organizational, and domain knowledge combined with private data such as email inboxes, browsing history or chat logs can be used for identity theft. For example, using the aforementioned pieces of information, a malicious actor can open bank accounts for money laundering.
- *Spear Phishing:* The knowledge of internal/private data in combination with organizational, and domain information can be used to orchestrate targeted phishing campaigns. Thus, increasing the susceptibility of the potential victims to perform the attacker’s desired action.

4.4.3 Limitations

We selected the dataset specifically to only repositories which contained the string "dotfiles" in the name or description. Therefore, there may be a wide range of other non-code repositories, even some that contain personal configuration files which are not captured by this process. There may also be a bias in the frequency of findings due to copy-pasted configurations, which are probably not even used. This is limited though, since we focus on vulnerabilities which are typically not copied intentionally as well as the exclusion of forks from our dataset. We excluded repositories, which were larger than one GB, but that only affected a few. We also only fetched data from GitHub, although there are other shared version-controlled hosters available (such as GitLab or BitBucket).

Also, we did not test any secrets or vulnerabilities we found (see “Ethics” in Section 4.4.6). Therefore we cannot tell for sure, that any credential or vulnerability is valid. We even have evidence to believe, that the biggest platforms automatically invalidate secrets that

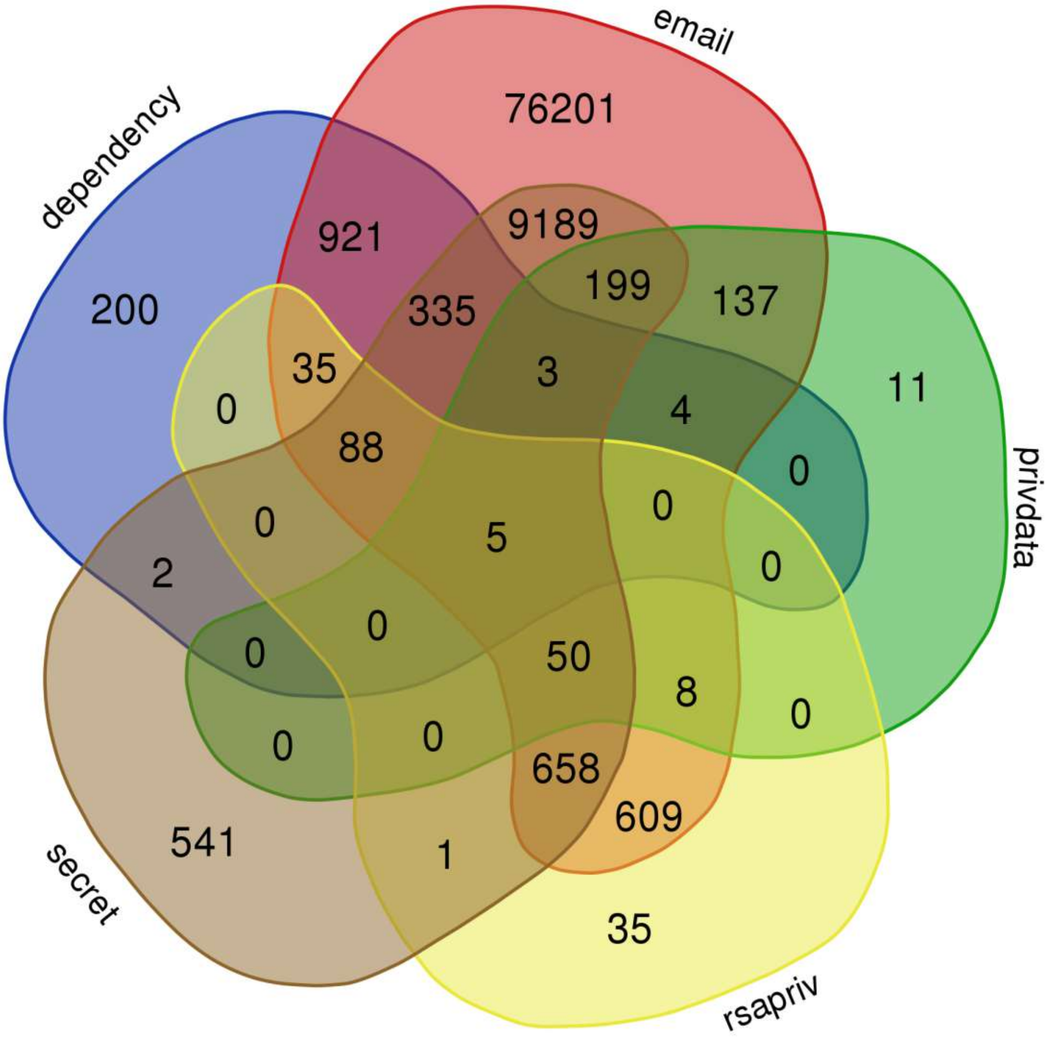


Figure 4.1: Venn Diagram of all available combinations of our findings per repository

are leaked through GitHub (see Section 4.3.1). We also did not focus on separating false positives from true positives, since that has already been covered by other research - among others with AI methods [36, 21].

Another inherent limitation comes from our approach of iteratively exploring the dataset. We provided a range of possible attack vectors, however, those scenarios are neither exhaustive nor indicative of all potential attacks.

4.4.4 Root Causes

Our iterative dotfiles analysis did not conclusively answer the question of what motivates people to share their dotfiles online and why sensitive data ends up in these repositories. We observed that an important aspect is convenience – the advantage of having version control and easy distribution of a repetitive task like configuring a new system to one's preferences. Since most configuration files in UNIX systems are text-only files, this is perfectly suitable for a VCS like git. Also, the tech community has advanced this trend through discussions, blog posts, tutorials, and helper scripts that automate the process of creating dotfiles and managing them. Many of these tools even address the issue of secrets and provide solutions (see Section 4.4.5). Nonetheless, there is a wide variety of tools available and many developers even prefer not to use any tools except for plain git. Based on these results, we decided to conduct the present survey to learn more about the motivation of developers to publicly share their dotfiles.

When secrets get added to a repository, they often are deleted incompletely by creating a new commit, without rewriting the history. Around 59 % of the leaks (on the example of API keys) we found were in the historic states of commits but not in the most recent state (HEAD), thus supporting our claim. This may be due to users being unaware of the secret they committed or insufficient knowledge how to delete sensitive data correctly from git. GitHub provides help for this task (see section 4.4.5), but it seems, this information has not reached everyone.

4.4.5 Recommendations

The use of dotfiles has many benefits to users, but everyone should think carefully about what to share and how. With the availability of unlimited private repositories in the free plan of GitHub, the first consideration should be, whether a private repository is the better choice.

Secondly, there is a very good collection of tutorials and tools provided by some users on GitHub[7], which developers should read into, before creating their own dotfiles repositories. The adoption of a well-known strategy and a popular tool may decrease the risk handling ones dotfiles in an insecure way.

Lastly, we noticed, that not everyone seems to know, how to delete information from a git repository safely. It is important to note that, once an information has been online, no matter how short, it should be considered compromised and appropriate steps should be taken. This is because any upload is immediately propagated to the Events API and also could end up in other places like GHTorrent swiftly as pointed out by Meli, McNiece, and Reaves [23]. Remediation actions include deleting the information from the git history, deleting the credentials from the repository and revoking the credentials.

[7] https://dotfiles.github.io/

GitHub's documentation on removing sensitive information from a repository provides several useful guidelines in these regards.[8]

4.4.6 Ethics

This section explains the ethical considerations that have been conducted for this research. First of all, we only worked with public data. All repositories we examined were available on GitHub to the public and have been found through the public API.

We also never tried to use any of the secrets and vulnerabilities we found through GitHub or test their validity. We therefore cannot guarantee the validity or up-to-dateness of the data but we can rule out any legal or personal consequences, such as red team testing would have.

Lastly, we responsibly disclosed the issues we found directly to those repository owners, which we had a contact from and all published information that stems from our dataset is aggregated or anonymized.

[8] https://docs.github.com/en/github/authenticating-to-github/removing-sensitive-data-from-a-repository

CHAPTER 5

Qualitative Research

5.1 Methodology

Since dotfiles are highly idiosyncratic and closely linked to individual owners, we wanted to get a deeper insight into the motivation for sharing them. This motivated us to conduct a survey to learn more about the motivations, security knowledge and considerations and develop conclusions and recommendations on the base of the results. We will describe the process and the findings in this chapter.

5.1.1 Understanding Dotfile-Users

In the previous chapter, we showed numerous potential attack vectors possible in the dataset containing GitHub dotfile repositories. These repositories contain personal configurations and other user-related data. Therefore they are tightly bound and unique to individual persons. This leads to the question of what motivates people to share their dotfiles and whether such data should be published openly at all. During our research, we found that people share and discuss their dotfiles on diverse platforms like specific communities on Reddit as well as other fora. In this chapter, we try to explore users' motivation and perceptions of sharing dotfiles on diverse platforms.

In order to get a representative and comprehensive answer on how and why people share their dotfiles and learn more about these motivations, we decided to conduct a survey. Broadly, we were interested in who the people are that publicly share their dotfiles on GitHub and why they do it (**RQ2**) and what their knowledge is, if any, of the attendant security and privacy issues (**RQ3**). In order to develop our survey, we first defined the following survey-related (sub-)research questions. The first two are about finding who and when and the third and fourth are about the motivation of sharing.

SRQ1 "Who" shares their dotfiles on GitHub?

SRQ2 Since when/for how long do they share them?

SRQ3 Why do they share it publicly?

SRQ4 Why do they share it in general?

SRQ5 What is their relationship with and competence in security and privacy (w.r.t dotfiles and in general)?

In the next sections, we want to explain, how we designed our survey and questionnaire to best answer these questions, mitigate possible biases and maximize the return rate.

5.1.2 Survey Design

We formulated the following general goals to which we intended to design our questionnaire:

- Inductive approach (grounded theory alike) to minimize validation bias
- Quantitative questions to get a "bigger picture" of context
- Demographic questions to find unknown correlations
- Open questions to provide respondents the freedom for new ideas / their opinions / etc. and minimize bias
- Similarity to other GitHub/developer surveys to aid comparability

As noted above, we intended to follow an approach that is similar to the grounded-theory methodology [37, 42]. This means that we conducted our survey with no preset theory in mind and generated our findings inductively based on open-ended responses. To achieve this goal, we carefully formulated the two main concepts (the reasons for sharing dotfiles publicly and plans for future changes) as open questions with as little presupposition as possible. We conducted open coding after receiving the responses in order to evaluate the results. This will be explained further in Section 5.1.3.

Participants

As for sample selection, we decided to use the full sample of available contacts because we expected a low return rate. This is based on existing knowledge about this mode of survey propagation – unsolicited mass email. Based on our dataset (about 124,230 repositories), we selected those users, who publicly set an email address in their profile and would therefore express agreement to be contacted via mail. In total, these were 44,472 email addresses. We personalized the emails by addressing everyone with their username. In addition, we utilized this opportunity to disclose any security or privacy vulnerability we found in these repositories. These disclosures were also personalized to the individual recipients: each user received one or more sentences according to the five categories we present in our investigative study part. The exact email text is listed in the appendix as well. We automatically flagged the filled surveys with the information on whether a user received a disclosed vulnerability or not. Apart from that the surveys were identical to these two user groups (those who received a disclosure and those who did not). With this sample selection strategy we minimize the sampling and coverage error as far as possible. The bias which may result from the fact, that we can only send emails to people from which we have a usable email address lies outside our control. To further explain this, users who do not provide a public email address in their GitHub profile are probably more security-aware or have bigger corporations at stake. They would have therefore answered the survey differently, which would have given us different results. This issue can be seen as a form of sampling bias.

Survey Structure

Our survey (see Appendix A) contains four sections and a total of twenty questions. The first section is about the usage of GitHub in general. The goal is to make participants comfortable, make them familiar with the survey-topic and question style, and allow for a comparison of our user-base with the general group of GitHub users. The questions are formulated as single-choice, multiple-choice, and one numerical text field asking for the number of own repositories.

The second section is on the main topic of our survey, i.e., the use of dotfiles. It consists of six questions. The first is a binary question of whether the dotfiles repository of the user is still actively used. This allows us to compare their self-assessment with the definition we chose for our analysis (namely that active repositories are those, that have been updated in the current year). The question about the first usage of dotfile repositories follows the same purpose (to cross-check this data with the age of dotfile-repositories in our quantitative dataset) and also allows to get insights into the trend we only saw in individual statements in online communities so far (a trend about collecting, sharing and discussing configuration collections as well as system setups). The next three questions – what are the platforms and tools used to store and manage dotfiles and what proportion of copy-pasted contents constitutes dotfiles – also follow the purpose of understanding the trend better and allowing for a comparison with our existing dataset. We want to get an in-depth insight, into the questions when this trend emerged and in what direction it emerged. The final question is an open-ended one, formulated as "Why did you share your dotfiles on GitHub?" This allows participants to provide qualitative responses about their motivation and allows us to get an unbiased response, and rigorous picture, and sheds light on the thoughts and opinions of the participants on the matter – why they started to publish their dotfiles to GitHub. By balancing formality and directness, we tried to be as neutral as possible while challenging them to give this question a serious thought.

The third section evaluates the knowledge and stance on security and privacy of the survey participants – in general and in terms of dotfiles. The first four questions are presented as seven-point Likert scales. This choice has the disadvantage of the central tendency bias, on the other hand, it avoids the forced choice and the likelihood of acquiescence bias. The seven-point scales allow for a more fine-grained spectrum in contrast to the five-point ones and give the participants more options apart from the extremes and the middle, so that we shall get a clearer picture. The last question is our second open-ended question:

> We found several security & privacy issues across dotfile repositories on GitHub. If you are affected, you have received an email from us with further information. With this knowledge, what are your planned changes to your repository?

This question has multiple purposes. First, we wanted to evaluate, whether our disclosure was successful and had the desired effect of raising awareness for existing security problems. Secondly, we wanted to give participants a choice to evaluate their own security considerations and express further details about them. This way, they are incentivized to take action, even on issues we didn't uncover yet or as a general means of precaution. As we will see in the results sections, the mention of an email (see exact question above) caused some irritation and misunderstanding.

The last section of the survey tries to capture the general demographics and software development-related information of the participants. These questions allow us to compare our target group to those of similar studies such as, general developer studies done by GitHub or Stackoverflow. For this reason, the question design tries to mimic that of existing surveys.

Pilot and Testing

We refined our questionnaire and formulation through some internal iterations as well as pilot studies with two users, who fall into the target group of dotfile users although they do not share their dotfiles in a public GitHub repository. In addition, we consulted an internal expert on empirical studies. This way we made sure that our questionnaire is easily understandable by the target group and the questions are perceived as intended. We incorporated the feedback from the pilot study into the final survey design before it was sent out to the 44,472 recipients.

5.1.3 Survey Analysis

Execution

We sent out our survey via a mass-mail provider using a subdomain of our institute as From-address. Out of the 44,472 emails, 98.1 % were delivered, the rest had invalid addresses or were rejected by the receiving email server. We received 1,650 (response rate of 3.78 %) responses in the time frame of about three months (Oct 2021 - Dec 2021). Therefore we had to design an exact process in order to evaluate especially the two main open questions in an objective and reproducible way.

Coding

The coding of the data was done inductively, inspired by grounded theory [42]. The idea being, no presumptions or theories were developed before analyzing the data. Therefore, the first iteration was done without any limits or guides. Whenever a coder came across a statement that did not fit into any category already developed, a new category was added. After that, a synthesis was done horizontally across the coders as well as vertically across the categories. Our coding method for the first cycle can therefore be categorized as "initial coding" with the concrete realizations of "in-vivo coding" as well as "descriptive coding". *Initial coding*, which is sometimes also called open coding is the first round of a coding process with the goal of finding important topics and codes to further examine in the next steps. This closely relates to our intention as described above. *In-vivo coding* is defined by extracting the codes directly and literally out of the source data. We noticed during the initial phase, that the answers themselves sometimes provide just the right categories to use. *Descriptive coding* is a broad term and expresses that the goal of the coding process is a general description of the content to be coded. [37]

The final approach we chose was to select a random sample of one hundred answers. This sample was then independently coded as a first iteration by four members of our research project using open coding as described above. After that, an interrater agreement was calculated and a final coding guideline was formulated out of the reduction of these results. With this guideline, the rest of the answers were coded only once by individual members of our research.

5.2 Results

In this section, we will first describe our respondents in terms of some general categories and then go into more detail about their answers and opinions with regard to their use of dotfiles and notion of security. We will have a look at interesting correlations as well as other novel and notable statements from our dataset of answers.

5.2.1 Who are the Owners of Dotfile-repositories?

The majority of our survey respondents is quite young. As presented in Figure 5.1, around fifty percent are in their twenties and another third are in their thirties. The majority – i.e. 88 % – identify as male, about five percent identify as "Other" and about three percent as female as seen in Figure 5.3. The most frequent countries of origin of our respondents (who chose to answer this question) are the United States, Germany and the United Kingdom. More details can be found in Figure 5.2. The majority of them describes their occupation as software development, while the second most mentioned occupation is student. About three out of four respondents have an undergraduate or postgraduate degree. These results are similar to other developer-surveys [41].

Most of our respondents use GitHub for private projects. A large group is also involved actively in open-source development (54 %). Also, half of them already contributed to a

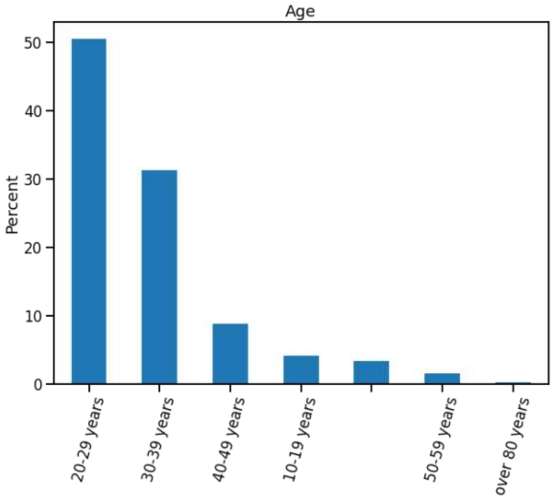


Figure 5.1: Survey results for the age of GitHub users

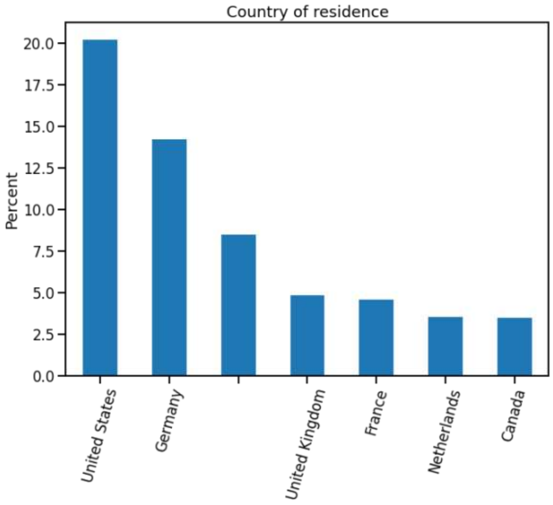


Figure 5.2: Survey results for the country of origin of GitHub users

project or fixed a bug. More details can be found in Figure 5.4. Many of them are quite active on GitHub and use it every day (49 %) or at least once a week (31 %), as seen in Figure 5.5. It is important, to keep the nonresponse bias in mind for this question. This means that active users are more involved and interested in the security and contents of their GitHub repositories and therefore – as we suspect – also more likely to respond to our survey.

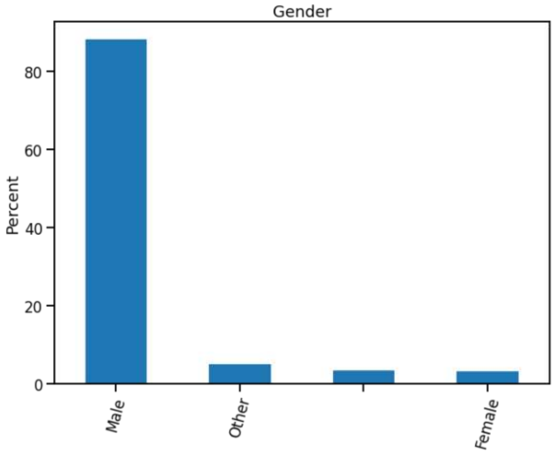


Figure 5.3: Survey results for the gender of GitHub users

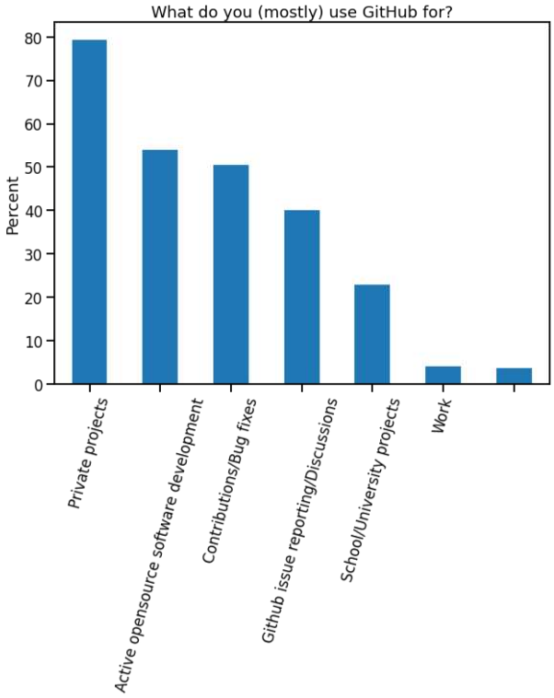


Figure 5.4: Survey results for the usage purpose of GitHub

5.2.2 Dotfiles Usage

As an extension to the previous question, we asked about the frequency and usage of the particular dotfiles repository. Four out of five respondents claim that they actively use the dotfile repository. This differs greatly from our assumption of active repositories. But it is essential to consider that active users are probably much more likely to answer such

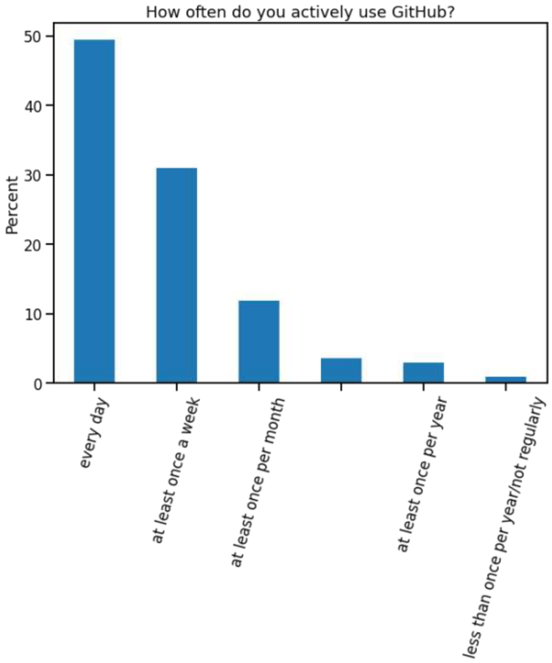


Figure 5.5: Survey results for the frequency of GitHub usage

a survey. More than half of them started to use dotfiles in the last five years and almost 90 % in the last ten years. We can therefore conclude that this trend has emerged in that timeframe, even though the concept of dotfiles themselves is very old. The majority uses only GitHub to share their dotfiles, but of course, this answer is biased since we targeted only GitHub users in our survey. Nonetheless, one-third of the respondents also reported some other sharing service. Out of those, the highest ranked answer was a private server with 16 %, followed by other online version control systems like Gitlab with 11 %. It is important to mention here, that also on this question, multiple answers were allowed. We were also interested in learning about participants' way of managing their existing dotfiles repository and what tools or technology they use, if any. The different responses and their percentages are summarised in Figure 5.6. A minimalistic approach to using basic git utilities for dotfiles management is the most popular method, followed by third-party management tools such as yadm, stow, dotbot, and chezmoi, which are also frequently discussed among the online communities. We also received additional open-ended responses where respondents mentioned using self-made tools and solutions to manage their dotfiles. Finally, in response to our question, *"How many (approx) of your dotfiles are self-written?"* most of our respondents claim that all or the majority of their dotfiles are self-written.

After the general understanding of participants' current usage methods of dotfiles, their sharing patterns and management approaches, the survey focuses on security and privacy-related questions. We collected the responses to our security-related questions in 7-point

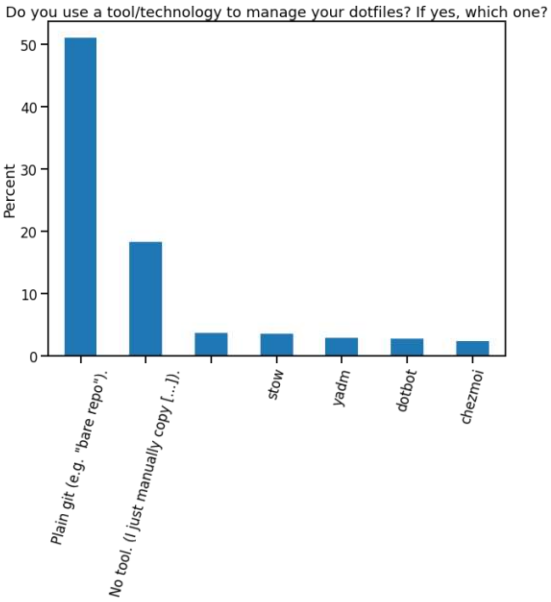


Figure 5.6: Survey results for the tool for dotfile management

Likert scales described in the previous section. The responses are visualized in Figure 5.7. We can see that the users rate the importance of security quite high and the actual security of their dotfiles repository, even though they are a bit modest about their own knowledge and competence on security.

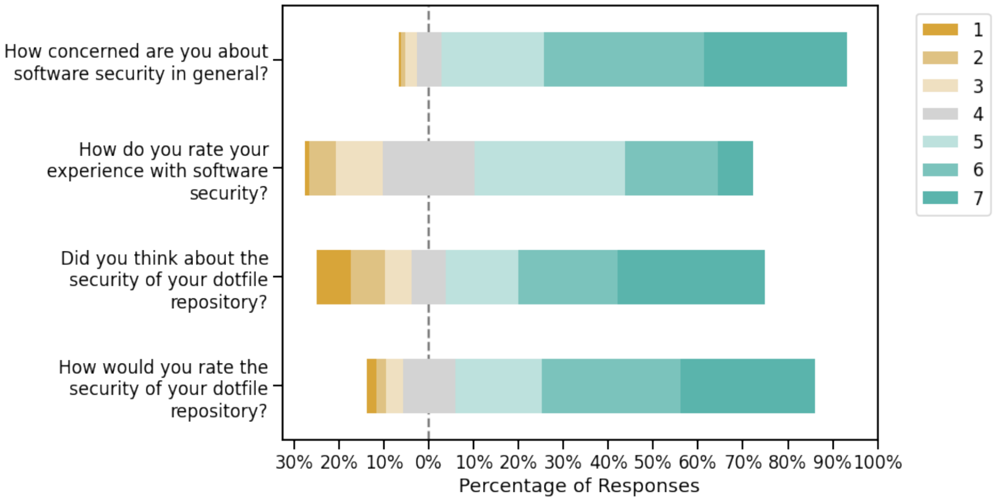


Figure 5.7: Security-related self-assessment based on survey

5.2.3 Motivation for using Dotfile Repositories

As described above, we intentionally asked about the motivation for using dotfile repositories as an open question. For our study, we were primarily interested in knowing about the motivation for using and sharing dotfile repositories. After two iterations of open coding and cross-validations between multiple coders (described in Section 5.1.3), we categorized and quantified the most common themes, and present them here.

Sharing (59 %) A majority of the respondents noted that they upload their personal dotfiles on GitHub and actively share them with others.

> *I often chat about my config with friends that are also into config management. Having it on GitHub makes it easy to show some lines while chatting.* – n0492

Many answers also emphasized a sense of community around personal configuration and learning from others.

> *Because we all learn from each other when we share.* – n1168

Setup (53 %) Another large share of respondents claimed that they used the repository to either quickly set up new machines or synchronize their configurations between physical machines and virtual machines.

> *So I can pull them from a new machine when setting up my dev environment.* – n0666

Backup (31 %) About one-third of the respondents also claimed they are using the repository particularly as a convenient storage or backup solution. This answer also includes people who are using it for its version control capabilities.

> *Reason number one was probably to have a cloud copy [...]* – n0117

Synchronization (23 %) About a quarter of respondents uses the dotfiles repository specifically to propagate changes between machines and operating systems. While synchronization and setup may sound similar, they distinctly emerged as two different themes in our data, so we decided to create separate categories.

Reference (9 %) Another class of responses argues about direct links in order to reference certain parts of their configuration. They use it for example when talking with friends or colleagues.

> *Mostly for quick reference when explaining how I do certain things on my system.* – n0894

Convenience (5 %) This category contains answers which emphasize the simplicity of using GitHub in general for storing or couldn't be matched to any other category (like: "for convenience").

> *Because it's convenient.* – n0705

No free private GitHub repositories (<1 %) Some people also mentioned that GitHub didn't have private repositories earlier - otherwise they would have made it private.

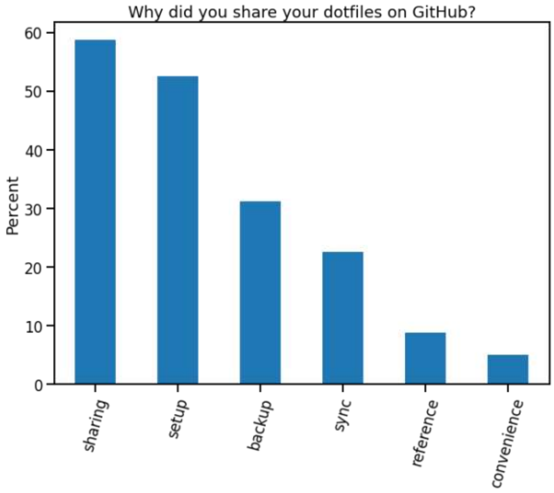


Figure 5.8: Survey results for the reasons for using dotfile repositories

A summary of the responses is shown in Figure 5.8. The survey responses show that a sense of community and open-source spirit is essential among the dotfiles repository users. The users believe in the ideology of sharing. They acknowledge learning by looking at others' configuration files, discovering best practices, tricks and eventually sharing their dotfiles, hoping others can benefit from it in the future. The second major reason stated by the survey takers falls under the broad category of convenience, where users like having their dotfiles in an easy-to-access location to help with backup and quick setup and synchronization between devices. Some respondents also mention that the lack of availability of private repositories made them switch to a public repository, thus making their configuration files publicly accessible. Furthermore, another set of respondents argued about the inconvenience of using a private repository and dealing with extra credentials, thus opting for the easiest way to share files between devices and platforms. These responses imply that people tend to choose convenient solutions over security and raise the question on how we can make secure solutions more usable.

Apart from the insights mentioned above, some of our participants also mentioned the term "ricing". The term "Rice" is commonly used to refer to making visual improvements and customizations on one's desktop. The community shares their nifty configurations and customizations of the default *NIX system to make it visually attractive on a Reddit group called r/unixporn [12]. This Reddit community currently stands at 366,560 ricers sharing and showcasing their configuration, aka dotfiles, with the community.

> *The dotfiles I share are simply cosmetic Linux configurations and/or small useful utilities, scripts, and customizations. There is a big community of "ricers" who like to show off these customizations, and so share and remix each other's dotfiles. I share mine because I want them in version control (I've lost them before), and so that folks who find aspects of them useful can use them.* – n0758

5.2.4 Possible changes

Finally, we asked our participants, what changes they plan to incorporate on their dotfile repository after reading our email and participating in our survey. We equally asked this question to participants which received a disclosure report about a possible privacy/security issue as well as those who didn't. After the coding process (cf. Section 5.1.3), we developed the following codes:

No change (58 %) More than half of the participants responded that they would not change anything in their repository. The participants were confident about the content of their files and considered it safe with no exposed vulnerabilities and security leaks.

> *I am careful to segregate sensitive information from configurations, so am fairly confident that I have not leaked anything.* – n0013

Check (5.8 %) Those who agreed to make changes claimed that they are mindful of committing sensitive information and make an effort to proofread the file contents regularly.

> *I routinely evaluate my dotfiles repo, and am not aware of any security risks beyond basic things like my email, editor preferences, etc.* – n0339

And those who received a vulnerability disclosure email from us claimed to look into the issues and take appropriate actions.

> *I will take a good look on what might be there that you found and remove it from all of that repositories history. Thank you for your project!* – l0041

Update (2.8 %) The third most frequent response was to delete any sensitive files and update the contents of their repository to make it safe while acknowledging the presence of sensitive contents.

> *The first thing I did was to delete my history backup file. Though it was a sqlite db file but anyone who had the deserializer that I was using, can get in plain text which contained a bunch of secret credentials.* – l0114

Delete repo (1.0 %) Some respondents went to the extreme end of stating that they will delete their entire repository from GitHub.

Make repository private (1.2 %) Another group of users claimed, they were going to make their repository private.

> *I changed its visibility to private. Before, githib didn't allow to go private for free.* – n1244

Tool (0.7 %) Interestingly for us, a small number of respondents also plan to use a tool to manage their dotfile repositories or the sensitive data within the repository.

> *Maybe encrypting my secrets in git repo (if any) with git-crypt* – n0182

From the survey results, we see that 58 % of respondents agree that they will not make any changes to the current state of their repository as we did not report any security or privacy issues pertaining to their repository, or they are reasonably confident about the content of their files and consider them safe. The second most common response was to *verify and update* the repositories. Here, the users were thankful for the research of identifying the security implications of openly sharing configuration files. The participants genuinely responded to review the contents of their files and update/delete any sensitive information leakage that might have occurred either due to carelessness or being unaware of the consequences of storing them in public.

> *Thanks for having my dotfiles repo scanned. Fortunately no leaks were found!It's a cool project and can help us, careless devs! Once in a blue moon I do some manual searching about such leaks and mail the owners in case I can find their email address. The last time I found some crazy stuff, such as bash_history files with inline mysql-client invocations including username, password and public server hostname! The sad thing is that about one out of twenty-thirty people do respond or take any action. The last one that replied to me, said he doesn't care! I hope you have fun with the project and crush your goal helping people fix their leaks.* – n0067

About 1.2 % of the participants responded that they would make the repository private. They claimed that the repository does not need to be public, and since GitHub now offers the creation of a private repository for free, they can safely switch to it. Apart from this, we also observed some other themes that fell into the category of *Additional Information* where participants asked for more details on the survey and the disclosed vulnerability; *Tooling* where a small number of respondents plan to use a tool to manage their dotfile repositories containing sensitive data such as git-crypt. Finally, *Unused* where 0.7 % of the respondents claim that they no longer actively use the repository and plan to delete it or make it private.

In summary, while most of the participants understand the security concerns related to shared configurations files, a decent number of participants do not plan to make any changes to their repository as they do not think of it as an issue or do not have any private information leakage in their dotfiles. On the other hand, it is motivating to see responses that benefited from our dotfiles research and urged the participants to be aware and mindful of committing sensitive data. Finally, we received a few critical responses on using non-privacy-preserving infrastructure such as Google Forms and personal email addresses for a mass survey. As mentioned, we included a proper ethics and disclosure statement on our survey and made sure only to use publicly identified email addresses from the recipients' profiles.

5.3 Discussion

5.3.1 Reasons for Sharing Dotfiles on GitHub

As we saw from these results, there are two main reasons, as for why people share their dotfiles on GitHub. The first group is the idealistic ones. They believe in the spirit of sharing - the spirit of the open-source community. They intend to show their results to friends, colleagues, or in online communities. Some mentioned specific boards on the online platform Reddit for that purpose. The answers, which we tagged as *reference*, follow a similar purpose. These include respondents who like to refer to a specific configuration when talking to someone or explaining something. This group is very well suited for the social and collaborative aspects of GitHub. The second group of explanations includes pragmatic reasons. For those users, it is convenient to have a centralized storage or backup for this kind of files. They take advantage of the benefits of version control. Many users also argued that this repository allows them to quickly set up new machines or synchronize configuration changes between running machines. For this second group, a private repository would probably be sufficient. But there are also some shortcomings with private repositories – one user responded for example, that they appreciate the convenience of downloading their configuration quickly on virtual machines without any authentication. This would not be possible with a private repository. But in general, a private repository is most likely the best option, and easily available since GitHub nowadays allows unlimited private repositories for free.

5.3.2 Relationship between Convenience and Security

We even got some explicit answers from participants saying, that if free private repositories were available at the time they started to use dotfiles, they would have used those. Another common statement was, that participants were planning to switch to a private repository after reading our email and participating in the survey. We, therefore, think that convenience is an important concept to consider for platforms to aid security. This means security-related features should be provided by platforms as conveniently as possible and free of charge.

5.3.3 Side-effects and Suggested Next Steps

As for the disclosure part of our email (and survey), we received interesting responses as well. Independently of whether users received it as part of a disclosure (or just the general information and the survey link), many of them were thankful for our dissemination and said they were planning to make some changes to their dotfiles repository. Out of those, the majority were planning to go through their files or update them individually. But still, 1.1 % is planning to make their repository private, and 1 % is planning to delete their repository from GitHub altogether. Furthermore, it is interesting, that 0.7 % said, they were going to use a tool to manage their dotfiles or check for security and privacy issues. This may be a good alternative since from the answers to other questions we see that not using any supporting tooling or using something self-written is by far the most common approach. As the community grows and evolves, we hope to see more tools that can strengthen the security and privacy aspects of personal configurations. These will also be able to provide more sensible defaults of what files to include and warnings if any keys or credentials are about to be committed.

5.3.4 Problems of Understanding

We also received a large number of responses per mail and in the survey answers that indicated different problems of communication. First of all, many of our respondents notified us about the fact that the issue we reported on their repository was a false-positive. We were aware of that already, but – as mentioned also in Section 4.3.1 – we focused on different aspects and therefore didn't take efforts to eliminate those.

Another clear issue of miscommunication was, that those who did not receive a disclosure, received the statement "No leaks have been found in your repository". Nonetheless, they were asked to fill out the same survey. In the survey, we asked about their changes in reaction to our disclosure. That is why many of them expected a second email with a disclosure or more details about it. We clarified the arising questions by setting up an information website and writing a response to the questions we received.

5.3.5 Limitations and Ethics

We sent our survey to owners of public dotfiles repositories on GitHub, who had set a public email address on their profile. This naturally limits the responses to exactly this user group. Users who are more privacy- or security-aware would not be contacted, because they may already use a private repository or do not have a public email address to contact them. This is a form of sampling bias which is also mentioned in Section 5.1.2.

Our university does not have an ethics board, therefore we followed our internal guidelines and procedures and are going to lay them out here. To contact our survey participants, we only used publicly available information - as stated above. The email has been sent once, and only those who specifically asked for further information or the results of our study have been contacted again to answer those requests. In the email they have been informed about the purpose of the survey and the usage of their data. We are never – and cannot – link individual answers to specific repositories. The answers we published, have been checked to not reveal any information that is directly linkable to a specific user.

CHAPTER 6

Conclusion

This study provided a comprehensive insight into the security and privacy implications of publicly sharing personal configuration files (dotfiles). We performed a large-scale analysis of 124,230 public dotfiles repositories on GitHub and discovered potential sensitive information leakage in 73.6 % of the analyzed repositories.

Since personal configurations are closely linked to a singular user's particular system – dotfiles repositories pose a highly personal security risk. In our research, we evaluated and classified several attack vectors such as credential stuffing, impersonation, or phishing due to direct and indirect security vulnerabilities identified in the dotfiles. We also surveyed 1,650 repository owners to understand their motivations, awareness, and perceptions of security and privacy risks. We found that sharing is mainly ideological (an end in itself) and to show off ("ricing"), and for providing a reference to other users. Our study also found that participants commonly use and share dotfiles for the convenience of machine setup, synchronization, and backups. Most users are confident about the contents of their files and claim to understand the security implications, and will continue sharing their dotfiles after taking appropriate actions.

In light of the popularity and widespread use of dotfiles repositories, participants should be able to make informed decisions about the security and privacy of their files. In that regard, we hope our research can help inform future standards, usage, implementations, and sharing of dotfiles repositories.

CHAPTER 7

Further Work

We recommend further work in the field of security and privacy issues of public data. This is necessary because the risk not only impacts individuals but also possibly larger entities like companies or open-source software projects. Therefore we suggest the development of more sophisticated automated tools like Gitleaks. Gitleaks and others already do a great job, when it comes to private keys or API credentials and a well integrated into automated checks by GitHub itself or within organizations. But they need to be extended to find non-trivial issues like the ones we described. There are already a lot of tools to find outdated dependencies for example. An automatic correlation of certain packages or configurations with new CVEs would be even better in that regard.

The survey we conducted specifically targeted users within our dataset. This had the advantage of finding correlations between what they responded to and our mining results. But it would be really interesting, to extend this study to active users e.g. in the Reddit Community, on Social Networks, or on other Platforms like GitLab or BitBucket. This would reveal a broader picture and eradicate the possible sampling bias we might have in our study.

We also want to emphasize the active mitigation strategies we recommend. On a direct user level, we recommend campaigns to raise knowledge about security and privacy issues of configuration data. This may be done directly in the community, e.g. with Reddit Posts and a Wiki entry. Also, the (private) dotfiles landing page on GitHub would be an opportunity to emphasize this information. Tools that sensibly manage one's dotfiles could be recommended as well as information on how to deal with compromised information. Also, tools that are specifically designed for private information like Gitsecret could be promoted.

On a platform level, there is on one hand the possibility of automated checks and issuing of warnings. On the other hand, sensible defaults can have a great impact. For dotfiles repositories, there are a few options: (1) the suggestion to make it private; (2) a

`.gitignore` file that only includes real configuration files and no databases containing caches, cookies, or passwords; (3) a template including the setup of one of the tools, mentioned above.

APPENDIX A

Survey Questions

Thank you for participating in our survey. The survey consists of 4 sections and will take about 5–10 minutes. All responses are anonymous. Our research is done by [anonymized for review]. It focuses on the security and privacy related aspects of shared configurations, a.k.a. "dotfiles". Your response provides valuable information and helps us formulate recommendations on the security of this domain for the open source community. Our findings will be published as a paper. If you want to send us additional feedback, concerns or want to get notified about the results please send us a message at [anonymized for review]

* *required*

Q1. What do you (mostly) use GitHub for?
○ Private projects
○ Active opensource software development
○ Contributions/Bug fixes
○ Github issue reporting/Discussions
○ School/University projects
○ Other: ___________

Q2. How many repositories of your own do you have on GitHub (self-created)?

Q3. How often do you actively use GitHub?
○ every day
○ at least once a week
○ at least once per month
○ at least once per year
○ less than once per year/not regularly

Q4. Do you still actively use the dotfile repository?
○ Yes ○ No

Q5. When did you first start to use dotfiles?
○ 0-5 years ago
○ 5-10 years ago
○ more than 10 years ago

Q6. Did you first/also share them in other ways/platforms — if yes, where?
○ No, I don't share dotfiles on other platforms
○ Dropbox
○ Other cloud file storage
○ Other cloud version control service (e.g. Gitlab, Bitbucket...)
○ Private server
○ Other: ___________

Q7. Do you use a tool/technology to manage your dotfiles? If yes, which one?
○ No tool. (I just manually copy my dotfiles to the right place).
○ Plain git (e.g. the "bare repo" approach).
○ dotbot
○ chezmoi
○ rcm
○ yadm
○ Other: ___________

Q8. How many (approx) of your config files are self-written and how many are copy-pasted from somewhere?
○ all are self-written
○ most are self-written
○ about half are copy-pasted, the other half self-written
○ most are copy-pasted
○ all are copy-pasted

Q9. Why did you share your dotfiles on GitHub?

Q10. How concerned are you about software security in general?
○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

Q11. How do you rate your experience with software security?
○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

Q12. Did you think about the security of your dotfile repository?
○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

Q13. How would you rate the security of your dotfile repository?
○ 1 ○ 2 ○ 3 ○ 4 ○ 5 ○ 6 ○ 7

Q14. We found several security & privacy issues across dotfile repositories on GitHub. If you are affected, you have received an email from us with further information. With this knowledge, what are your planned changes to your repository?

Q15. Age *

○ 10-19 years ○ 20-29 years ○ 30-39 years
○ 40-49 years ○ 50-59 years ○ 60-69 years
○ 70-79 years ○ over 80 years

Q16. Gender *

○ Female ○ Male ○ Other

Q17. Country of residence

○ ... list of countries ...

Q18. Highest educational degree

○ School, no diploma
○ Secondary education (high school)
○ Trade/technical/vocational training
○ Undergraduate education (college or university)
○ Postgraduate education (masters or doctorate)
○ Other: ___________

Q19. What is your current occupation?

Q20. How many years of experience do you have in software development (if any)?

APPENDIX B

Survey/Disclosure Email

Hello *[GitHub Username]*,

we are a research team at TU Wien, Austria. We are writing you, because you are using GitHub and have a repository with configuration files (dotfiles). We did research on the usage and security of these repositories.

We found the following issues with your repository (if any):

(No leaks have been found in your repository)

[OR 1 or multiple of the following:]

* Credentials: Your repository may contain API keys or authentication credentials, which(if valid) could be used to log in to web services in your name.
* RSA Keys: You may have a private key or weak public RSA key, which could be used to authenticate to some service(e.g. via ssh) in your name.
* Private Data: Your repository may contain private data, which is typically not shared publicly. This includes, browsing history, cookies, and chat logs.
* Old/Outdated Dependencies: Your repository may contain software dependencies, which are outdated or misspelled. These could, if installed somewhwere, contain security vulnerabilities.

In order to better understand how and why you use shared configurations, we designed a small survey. We would be very happy, if you filled it out. It takes about 10-15 minutes.

https://forms.gle/oJe9SWf1KU4MdbZV6

If you have any additional notes, questions or feedback, you can reply to this email.

Thank you for your time
Best regards
Gerhard Jungwirth (TU Wien)

List of Figures

List of Tables

Bibliography

[1] Stefan Achleitner et al. “Deceiving Network Reconnaissance Using SDN-Based Virtual Topologies”. In: *IEEE Transactions on Network and Service Management* 14.4 (Dec. 2017). Conference Name: IEEE Transactions on Network and Service Management, pp. 1098–1112. ISSN: 1932-4537. DOI: 10.1109/TNSM.2017.2724239.

[2] Abdulmajeed Alqhatani and Heather Richter Lipford. “"There is nothing that I need to keep secret": Sharing Practices and Concerns of Wearable Fitness Data”. In: Fifteenth Symposium on Usable Privacy and Security (SOUPS 2019). SOUPS’19. 2019, pp. 421–434. ISBN: 978-1-939133-05-2. URL: https://www.usenix.org/conference/soups2019/presentation/alqhatani (visited on 05/04/2023).

[3] Daniel V. Bailey, Philipp Markert, and Adam J. Aviv. “"I have no idea what they’re trying to accomplish" Enthusiastic and Casual Signal Users’ Understanding of Signal PINs”. In: Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021). 2021, pp. 417–436. ISBN: 978-1-939133-25-0. URL: https://www.usenix.org/conference/soups2021/presentation/bailey (visited on 02/16/2022).

[4] Mihai Barbulescu et al. “RSA Weak Public Keys Available on the Internet”. In: *Innovative Security Solutions for Information Technology and Communications*. Ed. by Ion Bica and Reza Reyhanitabar. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2016, pp. 92–102. ISBN: 978-3-319-47238-6. DOI: 10.1007/978-3-319-47238-6_6.

[5] Elaine Barker. *Recommendation for key management:: part 1 - general*. NIST SP 800-57pt1r5. Gaithersburg, MD: National Institute of Standards and Technology, May 2020, NIST SP 800–57pt1r5. DOI: 10.6028/NIST.SP.800-57pt1r5. URL: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf (visited on 05/17/2021).

[6] Alex Birsan. *Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies*. Medium. Feb. 9, 2021. URL: https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610 (visited on 06/05/2021).

[7] Hudson Borges, Andre Hora, and Marco Tulio Valente. “Understanding the Factors That Impact the Popularity of GitHub Repositories”. In: *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME). Oct. 2016, pp. 334–344. DOI: `10.1109/ICSME.2016.31`.

[8] Noah Bühlmann and Mohammad Ghafari. “How do developers deal with security issue reports on GitHub?” In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. SAC ’22. New York, NY, USA: Association for Computing Machinery, May 6, 2022, pp. 1580–1589. ISBN: 978-1-4503-8713-2. DOI: `10.1145/3477314.3507123`. URL: `https://dl.acm.org/doi/10.1145/3477314.3507123` (visited on 04/30/2023).

[9] Casey Casalnuovo et al. “Developer onboarding in GitHub: the role of prior social links and language experience”. In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, Aug. 30, 2015, pp. 817–828. ISBN: 978-1-4503-3675-8. DOI: `10.1145/2786805.2786854`. URL: `http://doi.org/10.1145/2786805.2786854` (visited on 05/03/2021).

[10] Ben Cox. *Auditing GitHub users’ SSH key quality*. June 2, 2015. URL: `https://blog.benjojo.co.uk/post/auditing-github-users-keys` (visited on 05/27/2021).

[11] Constanze Dietrich et al. “Investigating System Operators’ Perspective on Security Misconfigurations”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. CCS ’18. New York, NY, USA: Association for Computing Machinery, Oct. 15, 2018, pp. 1272–1289. ISBN: 978-1-4503-5693-0. DOI: `10.1145/3243734.3243794`. URL: `https://dl.acm.org/doi/10.1145/3243734.3243794` (visited on 04/30/2023).

[12] Jie Fang. *The Basics of Ricing Linux*. Apr. 2016. URL: `https://jie-fang.github.io/blog/basics-of-ricing/` (visited on 04/24/2016).

[13] Felix Fischer et al. “Stack Overflow Considered Harmful? The Impact of Copy Paste on Android Application Security”. In: *2017 IEEE Symposium on Security and Privacy (SP)*. 2017 IEEE Symposium on Security and Privacy (SP). ISSN: 2375-1207. May 2017, pp. 121–136. DOI: `10.1109/SP.2017.31`.

[14] Tanner Fry et al. “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*. MSR ’20. New York, NY, USA: Association for Computing Machinery, June 29, 2020, pp. 518–522. ISBN: 978-1-4503-7517-7. DOI: `10.1145/3379597.3387500`. URL: `https://doi.org/10.1145/3379597.3387500` (visited on 05/17/2021).

[15] GitGuardian. *State of Secrets Sprawl on GitHub - 2021 report*. GitGuardian Blog - Automated Secrets Detection. Mar. 9, 2021. URL: `https://blog.gitguardian.com/state-of-secrets-sprawl-2021/` (visited on 05/27/2021).

[16] GitHub. *Searching for repositories.* URL: https://docs.github.com/en/search-github/searching-on-github/searching-for-repositories (visited on 02/17/2022).

[17] Eirini Kalliamvakou et al. "The promises and perils of mining GitHub". In: *Proceedings of the 11th Working Conference on Mining Software Repositories.* MSR 2014. New York, NY, USA: Association for Computing Machinery, May 31, 2014, pp. 92–101. ISBN: 978-1-4503-2863-0. DOI: 10.1145/2597073.2597074. URL: https://doi.org/10.1145/2597073.2597074 (visited on 04/07/2021).

[18] Paul Kari. "What you need to know about the biggest hack of the US government in years". In: *the Guardian* (Dec. 15, 2020). Section: Technology. URL: http://www.theguardian.com/technology/2020/dec/15/orion-hack-solar-winds-explained-us-treasury-commerce-department (visited on 05/03/2021).

[19] Ankit Kariryaa et al. "Understanding Users' Knowledge about the Privacy and Security of Browser Extensions". In: Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021). 2021, pp. 99–118. ISBN: 978-1-939133-25-0. URL: https://www.usenix.org/conference/soups2021/presentation/kariryaa (visited on 02/16/2022).

[20] Ben Lazarine et al. "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).* 2020 IEEE International Conference on Intelligence and Security Informatics (ISI). Nov. 2020, pp. 1–6. DOI: 10.1109/ISI49825.2020.9280544.

[21] Sofiane Lounici et al. "Optimizing Leak Detection in Open-source Platforms with Machine Learning Techniques:" in: *Proceedings of the 7th International Conference on Information Systems Security and Privacy.* 7th International Conference on Information Systems Security and Privacy. SCITEPRESS - Science and Technology Publications, 2021, pp. 145–159. ISBN: 978-989-758-491-6. DOI: 10.5220/0010238101450159. URL: https://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0010238101450159 (visited on 04/29/2021).

[22] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. "Impression formation in online peer production: activity traces and personal profiles in github". In: *Proceedings of the 2013 conference on Computer supported cooperative work.* CSCW '13. New York, NY, USA: Association for Computing Machinery, Feb. 23, 2013, pp. 117–128. ISBN: 978-1-4503-1331-5. DOI: 10.1145/2441776.2441792. URL: http://doi.org/10.1145/2441776.2441792 (visited on 05/03/2021).

[23] Michael Meli, Matthew R. McNiece, and Bradley Reaves. "How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories". In: *Proceedings 2019 Network and Distributed System Security Symposium.* Network and Distributed System Security Symposium. San Diego, CA: Internet Society, 2019. ISBN: 978-1-

891562-55-6. DOI: 10.14722/ndss.2019.23418. URL: https://www.ndss-symposium.org/wp-content/uploads/2019/02/ndss2019_04B-3_Meli_paper.pdf (visited on 04/29/2021).

[24] MITRE. *ATT&CK*. Apr. 29, 2021. URL: https://attack.mitre.org/ (visited on 05/14/2021).

[25] Daito Nakano et al. “A Quantitative Study of Security Bug Fixes of GitHub Repositories”. In: *arXiv:2012.08053 [cs]* (Dec. 14, 2020). arXiv: 2012.08053. URL: http://arxiv.org/abs/2012.08053 (visited on 02/25/2022).

[26] Wynn Netherland and Adam Jahnke. *GitHub does dotfiles - dotfiles.github.io*. URL: https://dotfiles.github.io/ (visited on 06/05/2021).

[27] Marc Ohm et al. “Backstabber’s Knife Collection: A Review of Open Source Software Supply Chain Attacks”. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Ed. by Clémentine Maurice et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 23–43. ISBN: 978-3-030-52683-2. DOI: 10.1007/978-3-030-52683-2_2.

[28] Linux man pages. *file(1): determine file type*. URL: https://linux.die.net/man/1/file (visited on 02/17/2022).

[29] Gustavo Pinto, Igor Steinmacher, and Marco Aurélio Gerosa. “More Common Than You Think: An In-depth Study of Casual Contributors”. In: *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*. 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER). Vol. 1. Mar. 2016, pp. 112–123. DOI: 10.1109/SANER.2016.68.

[30] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. “Security and emotion: sentiment analysis of security discussions on GitHub”. In: *Proceedings of the 11th Working Conference on Mining Software Repositories*. MSR 2014. New York, NY, USA: Association for Computing Machinery, May 31, 2014, pp. 348–351. ISBN: 978-1-4503-2863-0. DOI: 10.1145/2597073.2597117. URL: https://dl.acm.org/doi/10.1145/2597073.2597117 (visited on 05/02/2023).

[31] Alexander Ponticello, Matthias Fassl, and Katharina Krombholz. “Exploring Authentication for {Security-Sensitive} Tasks on Smart Home Voice Assistants”. In: Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021). 2021, pp. 475–492. ISBN: 978-1-939133-25-0. URL: https://www.usenix.org/conference/soups2021/presentation/ponticello (visited on 02/16/2022).

[32] Huilian Sophie Qiu et al. “Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source”. In: *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE). ISSN: 1558-1225. May 2019, pp. 688–699. DOI: 10.1109/ICSE.2019.00078.

[33] Baishakhi Ray et al. "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*. FSE 2014. New York, NY, USA: Association for Computing Machinery, Nov. 11, 2014, pp. 155–165. ISBN: 978-1-4503-3056-5. DOI: `10.1145/2635868.2635922`. URL: `https://doi.org/10.1145/2635868.2635922` (visited on 05/04/2021).

[34] Zachary Rice. *Gitleaks*. URL: `https://github.com/zricethezav/gitleaks` (visited on 02/17/2022).

[35] Matthew A. Russell. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. Google-Books-ID: _VkrAQAAQBAJ. "O'Reilly Media, Inc.", Oct. 4, 2013. 448 pp. ISBN: 978-1-4493-6822-7.

[36] Aakanksha Saha et al. "Secrets in Source Code: Reducing False Positives using Machine Learning". In: *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*. 2020 International Conference on COMmunication Systems NETworkS (COMSNETS). ISSN: 2155-2509. Jan. 2020, pp. 168–175. DOI: `10.1109/COMSNETS48256.2020.9027350`.

[37] Johnny Saldana. *The Coding Manual for Qualitative Researchers*. Google-Books-ID: RwcVEAAAQBAJ. SAGE, Jan. 27, 2021. 441 pp. ISBN: 978-1-5297-5599-2.

[38] Raphael Satter, Christopher Bing, and Joseph Menn. "Hackers used SolarWinds' dominance against it in sprawling spy campaign". In: *Reuters* (Dec. 16, 2020). URL: `https://www.reuters.com/article/global-cyber-solarwinds-idUSKBN28Q07P` (visited on 05/03/2021).

[39] Michael Schröder and Jürgen Cito. "An Empirical Investigation of Command-Line Customization". In: *arXiv:2012.10206 [cs]* (Dec. 18, 2020). arXiv: `2012.10206`. URL: `http://arxiv.org/abs/2012.10206` (visited on 04/29/2021).

[40] Jeffrey Spaulding, DaeHun Nyang, and Aziz Mohaisen. "Understanding the effectiveness of typosquatting techniques". In: *Proceedings of the fifth ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies*. HotWeb '17. New York, NY, USA: Association for Computing Machinery, Oct. 14, 2017, pp. 1–8. ISBN: 978-1-4503-5527-8. DOI: `10.1145/3132465.3132467`. URL: `https://doi.org/10.1145/3132465.3132467` (visited on 06/05/2021).

[41] Stack Overflow. *Stack Overflow Developer Survey 2021*. Stack Overflow. 2021. URL: `https://insights.stackoverflow.com/survey/2021/` (visited on 01/15/2022).

[42] Anselm Strauss and Juliet M. Corbin. *Grounded Theory in Practice*. Google-Books-ID: TtRMolAapBYC. SAGE, Mar. 11, 1997. 296 pp. ISBN: 978-0-7619-0748-0.

[43] Blake E. Strom et al. *Mitre att&ck: Design and philosophy*. Technical Report. Publisher: The MITRE Corporation. The MITRE Corporation, 2020.

[44] Jason Tsay, Laura Dabbish, and James Herbsleb. “Influence of social and technical factors for evaluating contribution in GitHub”. In: *Proceedings of the 36th International Conference on Software Engineering*. ICSE 2014. New York, NY, USA: Association for Computing Machinery, May 31, 2014, pp. 356–366. ISBN: 978-1-4503-2756-5. DOI: 10.1145/2568225.2568315. URL: https://doi.org/10.1145/2568225.2568315 (visited on 05/03/2021).

[45] Nikolai Philipp Tschacher. “Typosquatting in Programming Language Package Managers”. Bachelor Thesis. University of Hamburg, Mar. 17, 2016.

[46] Sam Varghese. “iTWire - SolarWinds FTP credentials were leaking on GitHub in November 2019”. In: *Wired* (Dec. 19, 2020). URL: https://itwire.com/security/solarwinds-ftp-credentials-were-leaking-on-github-in-november-2019.html (visited on 05/03/2021).

[47] Bogdan Vasilescu, Vladimir Filkov, and Alexander Serebrenik. “Perceptions of Diversity on Git Hub: A User Survey”. In: *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. 2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering. May 2015, pp. 50–56. DOI: 10.1109/CHASE.2015.14.

[48] Andrei Warkentin. ““Now if we could get a solution to the home directory dotfile hell!”[11]”. In: *Linux Symposium*. 2012, p. 55.

[49] Jason Warner. *Thank you for 100 million repositories*. Apr. 2018. URL: https://github.blog/2018-11-08-100m-repos/ (visited on 02/25/2022).

[50] Dominik Wermke et al. “Cloudy with a Chance of Misconceptions: Exploring Users’ Perceptions and Expectations of Security and Privacy in Cloud Office Suites”. In: Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020). 2020, pp. 359–377. ISBN: 978-1-939133-16-8. URL: https://www.usenix.org/conference/soups2020/presentation/wermke (visited on 05/04/2023).

[51] Craig E Wills, Kirstin Cadwell, and William Marrs. “Customization in a UNIX Computing Environment”. In: Seventh System Administration Conference. Monterey, California, 1993, p. 9.

[52] Hasan Yasar. “Experiment: Sizing Exposed Credentials in GitHub Public Repositories for CI/CD”. In: *2018 IEEE Cybersecurity Development (SecDev)*. 2018 IEEE Cybersecurity Development (SecDev). Sept. 2018, pp. 143–143. DOI: 10.1109/SecDev.2018.00039.

[53] Scott Yilek et al. “When private keys are public: results from the 2008 Debian OpenSSL vulnerability”. In: *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. IMC ’09. New York, NY, USA: Association for Computing Machinery, Nov. 4, 2009, pp. 15–27. ISBN: 978-1-60558-771-4. DOI: 10.1145/1644893.1644896. URL: https://doi.org/10.1145/1644893.1644896 (visited on 05/27/2021).

[54] Yue Yu et al. “Exploring the patterns of social behavior in GitHub”. In: *Proceedings of the 1st International Workshop on Crowd-based Software Development Methods and Technologies*. CrowdSoft 2014. New York, NY, USA: Association for Computing Machinery, Nov. 17, 2014, pp. 31–36. ISBN: 978-1-4503-3224-8. DOI: 10.1145/2666539.2666571. URL: http://doi.org/10.1145/2666539.2666571 (visited on 04/27/2021).

[55] Yaqin Zhou and Asankhaya Sharma. “Automated identification of security issues from commit messages and bug reports”. In: *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ESEC/FSE 2017. New York, NY, USA: Association for Computing Machinery, Aug. 21, 2017, pp. 914–919. ISBN: 978-1-4503-5105-8. DOI: 10.1145/3106237.3117771. URL: https://doi.org/10.1145/3106237.3117771 (visited on 06/07/2021).