Connecting the `.dotfiles`: Checked-In Secret Exposure with Extra (Lateral Movement) Steps

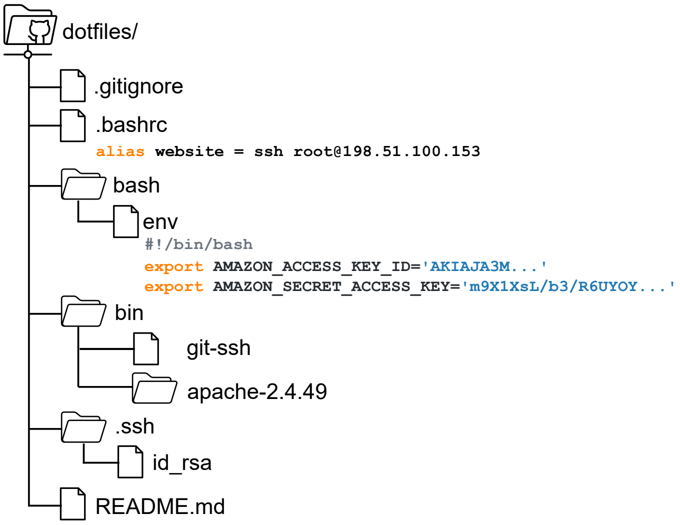
Gerhard Jungwirth,[*] Aakanksha Saha,[*] Michael Schröder,[*] Tobias Fiebig,[‡] Martina Lindorfer,[*] and Jürgen Cito[*]
[*]TU Wien, Vienna, Austria [‡]Max-Planck-Institut für Informatik, Saarbrücken, Germany

***Abstract*—Personal software configurations, known as *dotfiles*, are increasingly being shared in public repositories. To understand the security and privacy implications of this phenomenon, we conducted a large-scale analysis of dotfiles repositories on GitHub. Furthermore, we surveyed repository owners to understand their motivations for sharing dotfiles, and their awareness of the security implications. Our mixed-method approach consisted of two parts: (1) We mined 124,230 public dotfiles repositories and inductively searched them for security and privacy flaws. (2) We then conducted a survey of repository owners (n=1,650) to disclose our findings and learn more about the problems and implications. We found that 73.6 % of repositories leak potentially sensitive information, most commonly email addresses (of which we found 1.2 million), but also RSA private keys, API keys, installed software versions, browsing history, and even mail client inboxes. In addition, we found that sharing is mainly ideological (an end in itself) and to show off ("ricing"), in addition to easing machine setup. Most users are confident about the contents of their files and claim to understand the security implications. In response to our disclosures, a small minority (2.2 %) will make their repositories private or delete them, but the majority of respondents will continue sharing their dotfiles after taking appropriate actions. Dotfiles repositories are a great tool for developers to share knowledge and communicate – if done correctly. We provide recommendations for users and platforms to make them more secure. Specifically, tools should be used to manage dotfiles. In addition, platforms should work on more sophisticated tests, to find weaknesses automatically and inform the users or control the damage.**

I. Introduction

Credentials, API tokens, and other secrets accidentally committed to public software repositories, such as GitHub, are common security misconfigurations [30]. As such, accidentally published credentials and secrets have been the subject of recent research [25, 30, 32, 38, 54], as well as the initial point of compromise for major security incidents like the recent 'SolarWinds' hack of the U.S. Government [21, 41, 50]. Nevertheless, fixing security misconfigurations remains difficult [13], especially as it crosses the boundary to human factors, a field where *technical fixes* are usually not the solution to underlying issues [22, 53].

We develop the issue of secret and credential leaks from online software repositories a step further, taking context into account: We investigate the security impact of disclosed credentials and information in `.dotfiles`—files used to configure a user's environment on Unix and Linux systems—considering the additional context these files provide and how `.dotfiles` exacerbate the issue. The name `.dotfiles` comes from the 'dot' leading their path. On Unix and Linux systems, this leading dot excludes files from the output of `ls` if the `-a` parameter is not specified [52]. Hence, these files are commonly used to configure a user's environment. For example, `~/.profile`, a file in a user's home directory (`~/`), is automatically run by most shells when a user logs in (and starts their shell).



```
alias website = ssh root@198.51.100.153
```

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

Fig. 1: Structure of a dotfiles repository exposing sensitive information that *in combination* allows an attacker to compromise a system. An alias in the `.bashrc` file exposes the IPv4 address of a server on which the SSH key may work. Similarly, `apache-2.4.49` is an Apache version vulnerable to unauthenticated remote code execution (CVE-2021-41773).

However, `.dotfiles` are not limited to shell configuration files. Instead, on modern Unix and Linux systems, a user's complete environment—from Firefox to their email client and editor—is configured and customized via a variety of configuration files and databases in their home directory under `~/.*` (see Fig. 1 for an example). Thus, to improve the reproducibility of systems configurations, `.dotfiles` are increasingly version controlled and stored in stored in software repositories, such as GitHub. While this is a convenient way to store and share systems configurations, it can lead to unintended consequences with implications for security and privacy. As Fig. 1 illustrates, these files can not only leak credentials, but they *also* provide an attacker with valuable context on how and where these credentials can be used, or whether the `.dotfiles` repository's user is running outdated software. Note that this is not an artificial example: In fact, as recently as December 2021, the Dutch IRS experienced a full compromise by an ethical hacker due to an employee's `.dotfiles` repository [42]. In this case the context in the repository's `/.ssh/known_hosts` file provided additional information on *where* leaked credentials could be used.

In this paper, we investigate how the *context* provided by .dotfiles increases the security impact of leaked credentials and may lead to security and privacy issues, even if no *outright* secrets are published along with them. Furthermore, we investigate whether users are *aware* of the implications of sharing their .dotfiles, and why they decided to make their .dotfiles public. Specifically, we answer the following research questions:

RQ1 What are the security and privacy implications of sharing .dotfiles due to their additional *context*?
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 answer these questions, we first construct a taxonomy of information commonly shared in .dotfiles. We discuss how this information may threaten users' security and privacy, and how it provides additional context that increases the impact of leaked credentials. We then apply this taxonomy to a large-scale study of 124,230 .dotfiles repositories on GitHub, finding 10,942,606 potential issues in 73.6 % of all repositories, including 5,199 critical vulnerabilities. Then, to better understand *why* users share their .dotfiles, and whether they are aware of the implications of .dotfile sharing as-is, we conduct a survey among users (n=1,650) with .dotfile repositories.

In summary, we make the following contributions:

- We present a taxonomy of the security and privacy implications of .dotfile leaks, demonstrating that the issue of exposed .dotfile has implications beyond the exposure of credentials.
- Through a large-scale measurement study of 124,230 public .dotfile repositories on GitHub, we identify 10,942,606 security and privacy issues in 73.6 % of all analyzed repositories and personally notify 5,199 repository owners in a responsible disclosure process.
- Through a survey among repository owners (disseminated when informing them on our research and the security and privacy implications of exposed .dotfiles) we find that among our 1,650 respondents traditional explanations for misconfigurations [13], i.e., oversight and negligence, do not fully capture this phenomenon. Instead, exposure ties in deeply with users' personal vice and lacking risk-assessment (users are *aware* of risks, but do not consider them critical).

II. Security and Privacy Risks in .Dotfiles

We show in our research that .dotfiles are a sensitive source of public data on GitHub. Even though they are supposed to contain personal data, they are typically shared with the public intentionally. While the existence of API keys and credentials across GitHub repositories is well documented [28, 30, 39] and there exist several solutions, including GitHub's own Code Scanning,[1] we find secrets are still present at a large scale in .dotfiles repositories. In addition to these secrets, we found other forms of information disclosure that could be exploited by a malicious actor in different stages of an attack. Table I maps our findings to the relevant attack tactics and techniques in the MITRE AT&CK matrix [47].

TABLE I: MITRE AT&CK [47] classification of our findings.

Tactic ("why")	(Sub)-Techniques ("how")
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: Browser Credentials
	T1552: Unsecured Credentials
	.001: Credentials in Files
	.004: Private Keys

MITRE ATT&CK ("Adversarial Tactics, Techniques, and Common Knowledge") provides 14 categories of tactics adversaries use as part of an attack, i.e., their technical objective ("why"), and the corresponding (sub)-techniques they can use to achieve this goal ("how"). In particular, the tactics *Collection*, *Reconnaissance*, and *Credential Access*, and the relevant sub-techniques, align with our categories of exposed secrets in .dotfiles repositories. In particular, reconnaissance is an important phase of any contemporary, sophisticated attack [1]. Artifacts of interest include host identifiers (e.g., IP and MAC Address), email addresses, network information, or organizational information–information that we find potentially disclosed via .dotfiles. We also found information that leads to more direct attacks: aforementioned credentials, vulnerable software dependencies, as well as weak cryptographic keys. 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 Fig. 1). We further illustrate and enrich our taxonomy with quantitative data from our measurements (see Table III) and qualitatively tie results from our measurements to potential attack paths (see Section IV), demonstrating the impact of the exposed information in context.

In summary, due to their nature of including configuration files, .dotfiles repositories likely contain personal information, and can 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) in combination with host information to authenticate against the target services (e.g., cloud compute and storage services).
- *Vulnerable Packages:* Vulnerable packages can be identified from the package information stored in dependency files, 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.

[1] "GitHub Docs: About secret scanning" https://docs.github.com/en/code-security/secret-security/about-secret-scanning (last accessed April 2023)

- *Impersonation:* Personal, organisational, 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.

III. Measurement Methodology

In this section, we describe our methodology for measuring security and privacy issues in `.dotfiles` repositories on GitHub. We execute this pre-study prior to interacting with users who share dotfiles on GitHub to: (a) Identify the impact of secrets exposed in `.dotfiles`, i.e., to evaluate whether the additional context increases the security impact of exposed secrets (**RQ1**), (b) To have a foundation for creating our survey questions which relates to the *actual* use of `.dotfiles` repositories in the wild, and, (c) To validate that the issue of exposed repositories actually does exist at scale, and (d) to identify whether the GitHub user-base is an appropriate target for recruitment in our subsequent study.

A. Measuring `.dotfiles` Repositories

In total, we collected 124,230 dotfiles repositories including their version history from GitHub. To accomplish this, we developed a multi-step methodology that first identifies *public* `.dotfiles` repositories, and subsequently clones them.

Step 1: Identifying Repositories: In the first step, identify a list of repositories that potentially contain `.dotfiles`. For this purpose, we used the search function of the GitHub API [19] to search for variations of the string "dotfiles" (such as, Dotfile, dotFile, DotFiles, etc.) in either the name or description of repositories. We use these variations, as repository names do not necessarily contain the verbatim string "dotfiles," but instead it may be hyphenated, split, or adjusted. Furthermore, we did exclude forks from our search to limit space usage and avoid duplicates while gaining broader insight into the domain. We observed that forks generally use a comparable structure to their root repository. As such, we expect the quality of exposed files (in terms of use, credentials commonly contained, etc.) to stay consistent between a fork and its root. In about ten hours, the tool returned 125,171 repositories matching our criteria. The majority of the repository names (91 %) included the verbatim string "dotfiles."

Step 2: Downloading Repositories: To download the repositories locally, we used `git clone` with SSH public key authentication and a slight delay. We chose SSH for technical and automation reasons, but the same could be accomplished by unauthenticated cloning via HTTPS. In total, we collected 124,231 repositories out of that 125,171 were identified via the API. We excluded 56 of the remaining 940 repositories due to their size (more than 1 GB), while another 884 failed to download as they had been deleted or switched to private in the meantime. On average, each repository had 81 commits and three authors—although the number of authors per repository can be misleading, as the same person can commit to a repository using different authorship information, a problem that has been identified before [16] and was confirmed by us through manual inspection.

Step 3: Descriptive Analysis: After cloning all repositories, we collected general repository metadata and stored it in an SQLite database on an access-controlled server. We indexed all files present in each repository's working copy (i.e., the most recent version) and determined their size and, in the case of text files, their number of lines. Using the UNIX `file` utility [27] we also determined each file's MIME type.

Dataset Descriptives: In total, we analyzed more than 20 million files and found that the majority (61 %) were text files, such as text/plain, text/x-shellscript, application/json (see Table IIa). The most common file name was—naturally—`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 IIb).

Step 4: Security Analysis: In the final step, we automatically analyzed the downloaded repository data based on our taxonomy in Table III and the MITRE AT&CK framework [47] to identify potential security and privacy threats to answer **RQ1**. As the general identification of secrets in public Git repositories has already been covered by related work [28, 30, 39], we decided to extend existing open-access methodologies, specifically Gitleaks [37] to our research method. Gitleaks implements a methodology that recursively iterates over repositories and their full history, matching secrets via a pre-supplied list of regular expressions. We sampled our dataset and iteratively identify further relevant types of information that are common in `.dotfiles`. Subsequently, we crafted regular expressions for each of them and integrated them into Gitleaks to identify them in our dataset at scale. The full list of our newly added regular expressions can be found in Appendix A [20].

B. Ethical Considerations

Similar to all large-scale measurements, and security-focused research in particular, we need to consider the ethical implications of our work. TU Wien only recently established a Pilot Research Ethics Committee (Pilot REC), which reviews research plans prior to their execution. Our study predates the establishment of the Pilot REC, hence, we decided to closely follow the established best practices of the Menlo Report [3] independently. Specifically, we considered the following aspects and took the associated precautions:

- **Participation/Use of Public Data:** As we are conducting a large-scale evaluation of public data, the creators of repositories analyzed as part of our study can not individually consent to us using their published repository for our research. Hence, we conducted a harm-benefit analysis concerning the impact in *individuals* to assess the ethical aspects of our work. We concluded that: (1) The data we are using is public, hence all security issues we find may have been found by malicious actors as well. As we notified all cases of serious security issues if we could identify a point of contact, and do not utilize the data for objectives outside of the scope of our work, we consider our measurements as providing greater benefit than potential harm to users.

TABLE II: Overview of the most common file types and names, their frequency, and size.

(a) Most common MIME types.

MIME type	# Files	% Files	Average Size	Average # 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

(b) Most common file names.

Name	# Files	# Repositories
`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

(2) Our approach does not fundamentally differ from the approach taken by GitHub itself with Copilot when it comes to automatically analyzing a large set of repositories. As, contrary to Copilot, our motives are research driven and – see above – about helping users accidentally publishing secrets, we also consider our work ethical in this regard.

- **Storage and Access to Data:** Even though data may be publicly available, having it collected in a specific space may increase the chance for abuse. Hence, we stored data on a dedicated system only reachable from within our research institution. After the conclusion of the project, we deleted all collected data.
- **Validation of Vulnerabilities:** We did not attempt to exploit any of the found vulnerabilities and did not test the validity or recency of any exposed credentials. While such an approach might have lead to additional insights, we considered it as ethically unfeasible, as its impact would stand in no relation to the gained insights.

Considering our harm-benefit analysis and additional precautions we took to protect the collected data, we concluded that our research approach is sufficiently ethical.

IV. Measurement Results

In this section, we summarize the findings of our large-scale measurements and review them in the context of Section III. Furthermore, we illustrate non-intuitive attack pathways enabled by the information revealed in `.dotfiles`.

A. *Exposed API Keys*

A well-known problem of public repositories is that authentication credentials are accidentally committed into them [28, 30, 39]. These credentials can be passwords, API keys/tokens, or asymmetric private keys. Using our methodology, see Section III, we identified possible leaks of API keys in 11,758 repositories (9.5 %). Table IV gives an overview of all the credentials found in our scan, including asymmetric private keys. The most common leaked credentials in `.dotfiles` repositories are GitHub API keys, followed by Twitter Client IDs. To some extent, these results represent the general popularity of the different platforms.

Our results also include a limited number of false positives, evident by credentials that include strings such as "dummy" or "test." These only accounted for 6,294 (4.73 %) of all matches. As our study is explorative of the underlying issues and does not focus on exact measurement, we accepted this discrepancy and excluded affected keys from our analysis instead of using toolchains to more accurately identify these false positives, including approaches based on machine learning [28, 39].

The major difference to earlier work, focusing on repositories in general, is that the most commonly found API keys there were of Google cloud products [18, 30]. We trace this difference back to our focus on `.dotfiles` containing personal systems configuration, instead of code repositories in general. Hence, exposure of Git-related tokens is more likely. The crucial point here is that the leaked repository itself already provides context for the GitHub API key to be exploited efficiently.

Compromization Opportunity: A GitHub API token exposed in a `.dotfiles` repository allows – with high likelihood – an attacker to commit files to the repository itself. Especially if the user auto-deploys `.dotfiles` from their repository (something that can also become apparent from the repository) an attacker gains the ability to force the user to execute specific code, for example by editing the `.bashrc` (or similar shell configuration file) used by the user. Hence, contrary to traditional credential leakage which is usually limited to the remote project the key is tied to, leaked git credentials in `.dotfiles` repositories potentially lead to a full compromise of the user's environments due to their additional context.

B. *Leaked SSH Keys*

In our study, we found 9,452 private keys and an additional 3,145 public keys (2,844 RSA, 192 DSA, 109 ECDSA). Exposed private keys are an obvious problem, while *technically* public keys should be shareable [4] and GitHub itself even provides access to all public keys of its users. However, in both cases, the additional context of exposed `.dotfiles` repositories changes the threat landscape.

Compromization Opportunity: For private keys, `.dotfiles` make various targets and users where the private key might be used apparent. Hence, while a leaked private key for a system hidden behind several jump-hosts (SSH servers through which a user has to establish a connection) may be limited in impact, through security by obscurity, nonetheless, this obscurity is lost in a `.dotfiles` repository. Instead, an attacker may find a readily available SSH config pointing out the exact system and jump-host path a private key can be used for.

TABLE III: 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,
GitHub	65,589	6,898 (5.51 %)		Impersonation,
Twitter	38,752	3,936 (3.14 %)	20,760 across GitHub [30]	Spamming
Other	19,166	4,470 (3.57 %)		
RSA Keys				Hijacking,
Private Key	9,452	1,489 (1.19 %)	158,011 across GitHub [30]	Spamming
Public Weak Key	111	n.a.	Key length ≤ 1024 bit	
Public Vulnerable Key	6	n.a.	Debian RNG attack [55]	
PII				Spamming,
Email Addresses	1,227,175	88,442 (70.7 %)		Phishing
Private Data				Hijacking,
Firefox Logins	40	29 (0.02 %)		Impersonation
Thunderbird Profiles	2	2 (0.002 %)	Actual user data, not metadata	
Mailboxes	52	52 (0.04 %)	From Thunderbird and Mutt	
Software Packages				Hijacking
Python Dependencies	16,315	1,036 (0.83 %)		
JavaScript Dependencies	145,050	585 (0.47 %)		

TABLE IV: Number of matched Gitleaks rules.

#	Gitleaks rule
65,589	GitHub API Key
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	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

Similarly, the exposure of public keys along with systems they may be in use for enables various additional attack pathways for attackers. In our sample, 111 RSA keys had a key length of ≤ 1024 bit, which is a security risk [5]. Similarly, we still found six keys vulnerable to insecure random number generator (RNG) attacks [55], known for over a decade, by comparing them to a dataset of known affected keys [17]. In both cases, the considerations on exposed private keys above still apply. In addition, the use of old or outdated key material may (even if the keys are not publicly broken) indicate that a system has not been maintained with the appropriate care, and software running there may be as outdated as the recommendation to use ≤ 1024 bit SSH keys.

C. Software Packages

`.dotfiles` repositories often contain dependency graphs and other information revealing the state of dependency management on a users' machine. By analyzing these, we can infer that particular users are using certain software packages. In total, we were able to identify dependency trees in 1,621 repositories (1.3%). In these, the main source of structured dependency information comes from Python and JavaScript dependency definitions. Furthermore, we cross-checked all found dependencies with the publicly available package archives of the respective languages, and used the Levenshtein distance [12] to find similar packages for those which could not be found in the archives.

Compromization Opportunity: In-depth knowledge of the dependency tree used by a developer or organization enables several supply chain attacks [33], specifically *typosquatting* and *dependency confusion*. For typosquatting, in-depth knowledge about the utilized dependencies allows an attacker to carefully target their attack on a specific developer. Furthermore, if combined with write access to the repository, a typosquatted dependency may be stealthily injected into a user's `.dotfiles`. Due to the nature of typos, this compromise may go unnoticed for an extended period of time [44, 49].

Dependency confusion attacks [7] leverage the fact that package managers often prefer the highest version of a dependency. Hence, if a user uses a mix of internal and external repositories to source their code dependencies, an attacker may submit a version of a dependency that is usually only available via the *internal* dependency manager to an also used *public* repository. If the package in the public repository has a higher version number, the package manager will automatically install it. In practice, these attacks are extremely difficult, as it is often unknown whether a user uses a mix of internal and external repositories, and the names of internal packages are usually unknown. `.dotfiles` repositories, however, provide this additional context, significantly reducing the effort necessary for executing such a supply chain attack.

D. Personal and User Data

Personal and user data is a major issue with `.dotfiles` repositories, as on Unix and Linux common tools like chat programs, web browsers, and email clients write their content and logs to a user's home directory. If a user now blindly synchronizes their local .dotfiles with a remote repository, these files may become public. In our study, we discovered 52 mailboxes. Most likely, these commits were unintentional—we found that most of the mailboxes were in the historic states of the repositories and users attempted to delete them without overwriting the repository history. We also found private data from web browsers and chat programs, including caches, cookies, and browsing history.

Compromization Opportunity: Mailboxes and chat histories provide attackers with an abundance of data to use in a variety of attacks. This ranges from social engineering attacks [23] to credentials and account information found in email accounts. Depending on the organization, an email account may even contain messages including passwords and other credentials. Furthermore, in the context of the aforementioned attacks, communication logs may provide additional *context* to refine targeted attacks on an individual or organization.

V. User Survey Methodology

As outlined in Section II, exposing one's .dotfiles can have severe security implications if sensitive files are accidentally included. Nevertheless, sharing and showcasing one's code and configuration is an important aspect of open source culture [9, 14, 31, 56], enabling others to build and improve on existing work. Hence, we need deeper insights into the personal motivation for sharing .dotfiles publicly (**RQ2**), and users' knowledge regarding potential security and privacy implications (**RQ3**), in order to be able to suggest interventions and improvements to *enable* users to *securely* share their .dotfiles.

A. Questionnaire Design

In the design of our questionnaire, we broadly followed established best practices for questionnaire design [36]. To better capture specific nuances in .dotfiles repository use, we opted for a mixed methods approach. Our survey (see Appendix B [20]) consists of twenty questions across four sections, with a mix of single-choice, multiple-choice, Likert-scale, and open-ended answer types.

The first (Q1-Q3) and last (Q15-Q20) sections contain general questions about GitHub usage and demographic information, allowing us to compare our target group to participants in other GitHub based studies and with the general group of GitHub users. Furthermore, it serves as a validation question, to ensure participants are actively using GitHub. The second section (Q4-Q9) inquires about participants' use of .dotfiles repositories and contains questions to help us understand the trend of publicly shared configuration files. Furthermore, to capture more nuances and personal preferences concerning the use of .dotfiles repositories, it includes the open-ended question: *"Why did you share your dotfiles on GitHub?"* (Q9). This allows participants to provide qualitative responses that are not restricted by a prior questionnaire framework. By balancing formality and directness, we tried to be as neutral as possible, while challenging participants to give this question serious thought. The third section (Q10-Q14) contains questions to evaluate participants' knowledge of and stance on security and privacy issues, also in relation to exposed .dotfiles repositories. It consists of four questions on a seven-point Likert scale, which has the disadvantage of the central tendency bias, but on the other hand, avoids forced choice and reduces the likelihood of acquiescence bias [24, 35]. The final question is again open-ended: *"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?"* With this question, we want to explore whether our disclosure has an impact on raising awareness for existing security problems. Furthermore, we want to give participants a choice to evaluate their own security considerations and express additional details about them or considerations we might have missed. This way, they are incentivized to take action, even on issues we did not uncover or as a general means of precaution.

The questionnaire we ultimately used has been refined through internal iterations and a pilot study with two users, who fall into the target group of .dotfiles users. We iteratively incorporated the feedback from the pilot study into the final survey design before we sent it out.

B. Participants and Recruitment

We decided to recruit participants based on their having a publicly accessible .dotfiles repository with an associated contact email address. For our survey, we send an invitation email (see Appendix C [20]) to the full sample of 44,472 repository authors for which we could identify contact details in our dataset from the quantitative study. Please see Section V-D for a discussion of the ethics of this approach. We personalized the emails by addressing everyone with their GitHub username. In addition, we used this opportunity to disclose any security or privacy vulnerabilities we found in the recipients' repositories. Each recipient got an identical survey, independent of whether or not their repository contained a vulnerability. While this again limits the depth of our data, we considered it necessary to further preserve participants' anonymity.

We sent out a link to our survey (implemented with Google Forms) via Mailgun, a benign bulk-mail provider, using a subdomain of our host institution as the 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 responses to our survey (response rate of 3.78%). Our participants did not receive compensation for their participation, which would also have been difficult due to our anonymous data collection. We did not use other incentives like a raffle.

C. Evaluation

We use descriptive statistics to evaluate the nominal, ordinal, interval, and ratio data from the quantitative sections. We did not execute a deeper statistical evaluation, as the major objective of this data is to support and enrich observations from our qualitative questions. To evaluate the two open-ended questions in a qualitative and reproducible way, we used standard inductive coding techniques [40, 46]. Specifically, we drew a random sample of 100 answers of each question for an initial open coding round. In this first iteration, four members of our research group each independently developed codes for all answers, 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. Afterward, the four coders jointly synthesized the results horizontally across the coders and vertically across the categories and, resolving disagreements until consensus was reached, formulated final coding guidelines [29]. With this guideline and the final codes fixed, individual research group members coded *all of the remaining responses* in our survey.

D. Ethical Considerations

As already mentioned in Section III-B, due to the lack of an ethical review board at our institution at the time of our measurements, we followed established best practices for conducting human subject studies ourselves. Specifically, we designed our questionnaire to be fully anonymous and used ranges instead of exact numbers where personally identifiable information (PII) might be concerned. Furthermore, we informed participants that participation in the study is voluntary and they may stop at any time. Concerning our recruitment method, there are significant discussions on the viability of mass-recruitment via published addresses on, e.g., GitHub. In general, we followed the practices of earlier studies using GitHub user populations [8, 11, 26, 34, 43]. Given the objective of our work, we hence considered our recruitment efforts to be ethical in terms of a cost-benefit trade-off.

E. Threats to Validity

We invited active users of `.dotfiles` repositories on GitHub to participate in our survey. These participants were self-selected, i.e., the population we sampled is from users who actively use GitHub, and had the time and motivation to answer our survey questions (self-selection bias [6]). Users that did not respond to our survey may be systematically different from the rest of the population (non-response bias [2]). We may have also introduced bias through our survey questionnaire design. While using questions with Likert-type scales is an accepted instrument to assess levels of agreement, they are susceptible to acquiescence response bias [35], i.e., the tendency for respondents to agree with agree-disagree questions. We try to reduce the likelihood of this bias by using a seven-point Likert scale, which avoids forced choices.

VI. Survey Results

Here, we present the findings of our qualitative study. We first give a high-level overview of our respondents' demographics and then provide detailed results on their answers.

A. Survey Demographics

The majority of our survey respondents (50 %) are between 20-29 years old, 33 % are between 30-39. The majority (88 %) identify as male, about five percent identify as "other" and about three percent as female. Our respondents' most frequent countries of origin (who chose to answer this question) are the United States, Germany, United Kingdom, France, Netherlands, and Canada. Most of them describe 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 working with samples from GitHub [45].

B. Usage of `.dotfiles`

As a part of our survey, we asked participants how often they use GitHub and to select one or more reasons for using GitHub. Most of our respondents host private projects on GitHub. A large group is also involved in open-source development (54 %). Moreover, about half of the respondents contributed to a project or fixed a bug. Regarding the frequency of GitHub usage, 49 % of our respondents use GitHub daily, and 31 % use GitHub at least once a week, thus accounting for a decent share of active users in our survey. This proportion of active users is important to address the non-response bias, as active users are more involved and interested in the security and contents of their GitHub repositories and, therefore – as we suspect – are more likely to respond to our survey.

As an extension to the previous question, we asked about the frequency and usage of the particular dotfiles repository. Four out of five respondents in our survey claim that they actively use the specific `.dotfiles` repository. More than half of them started using dotfiles in the last five years, and almost 90 % of the respondents have been using it for the last ten years. In addition, we asked the participants to disclose other platforms and services they use for sharing their `.dotfiles`. The majority claimed to use only GitHub, although this number might be biased as our survey only targeted GitHub users. Nonetheless, one-third of the respondents reported using other sharing services and platforms such as a *private server* (16 %) or *other online version control systems* like GitLab and BitBucket (11 %).

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 summarized in Figure 2. 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 online communities. We also received additional open-ended responses where respondents mentioned using bespoke 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 seven-point Likert scales described in the Section V. The responses are visualized in Figure 3. We can see that the users rate the importance of security quite highly and the actual security of their `.dotfiles` repository, even though they are a bit modest about their own knowledge and competence in security.

C. Motivations for `.dotfiles` Repositories

For our study, we were primarily interested in the motivations for using and sharing `.dotfiles` repositories. After two iterations of open coding and cross-validations between multiple coders (described in Section V), we categorized and quantified the most common themes, and present them here.

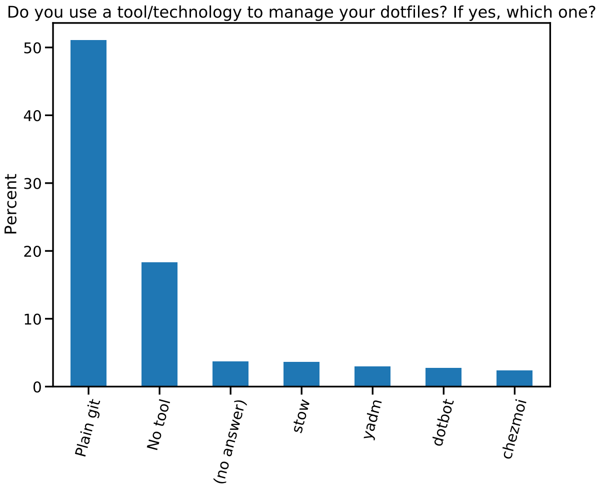


Fig. 2: Tools used for `.dotfiles` management.

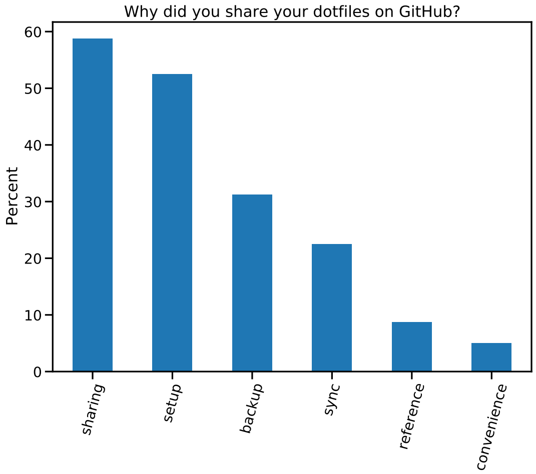


Fig. 4: Reasons for using `.dotfiles` repositories.

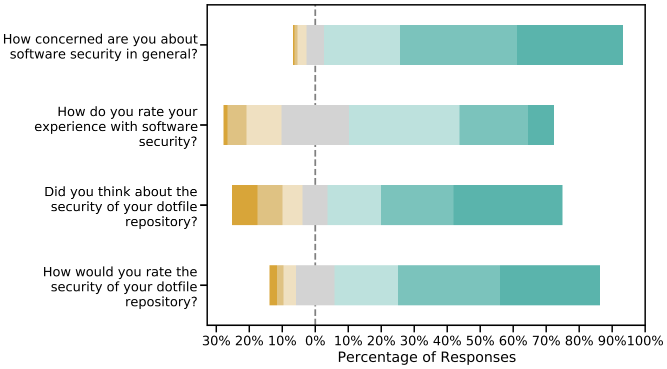


Fig. 3: Security-related self-assessment based on survey.

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, 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 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 use the 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

A summary of the responses is shown in Figure 4. 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 respondents 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 of how we can make secure solutions more usable. Apart from the insights mentioned above, we discovered a newish concept called “ricing” [14] as part of our investigation. The term “Rice” is commonly used to refer to making visual improvements and customizations on one’s system. The community shares their nifty configurations and customizations of the default *NIX (Usenix or Linux) system to make it visually attractive on a Reddit group called r/unixporn [14]. 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

D. Post-disclosure Interventions

In this section, we report on the participants’ response to the final open-ended question, which asks them to evaluate the security and privacy considerations and enlist any changes that they plan to incorporate in their repositories going forward. Here, we identified the following major themes:

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 content of their repository to make it safe while acknowledging the presence of sensitive content.

> *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, github 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 `.dotfiles` repositories or the sensitive data within the repository.

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

We also observed that 58 % of respondents say 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 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 configuration 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. On the other hand, it is motivating to see responses that benefited from our 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 in our survey and made sure only to use publicly identified email addresses from the recipients' profiles.

VII. Discussion

Limitations. In this section, we discuss the limitations of our methodological approach and results. Due to GitHub users copy-pasting their configurations across projects, there is some likelihood of inadvertently inheriting secrets from the initial source that were not directly introduced by the users of the repositories, thus, resulting in duplicate results. To limit the impact of such a workflow on our dataset, we excluded repositories that are forks of other repositories. Furthermore, in our research, we only mined data from GitHub, and do not include other publicly available code sharing platforms such as GitLab and BitBucket. Also, we did not test any secrets or vulnerabilities we found (see Section III-B). Finally, in Section II, we covered a range of possible attack vectors, however, those scenarios are neither exhaustive nor indicative of all potential attacks.

Recommendations. Although `.dotfiles` have many benefits to users, developers should be cautious of committing sensitive information to publicly available repositories. They should consider using the GitHub's private repository feature, which is currently free. There are also several tutorials and tools provided by GitHub[2] to help developers securely implement and maintain their repositories. It is important to note that, once a piece of information is online, no matter how short-lived, it should be considered compromised. The uploaded information immediately ends up on GitHub's Events API, which is publicly readable, as shown by Meli et al. [30]. As a result, users should take the necessary remediation steps, which include deleting the information from the repository's history, deleting the credentials from the repository, and revoking the credentials. GitHub's documentation on removing sensitive information from a repository provides several useful guidelines in these regard.[3]

Reasons for Publicly Sharing `.dotfiles`. As we saw from the survey results, there are two main reasons why people share their `.dotfiles` on GitHub. The first group are the idealistic ones. They believe in the spirit of sharing, i.e., the spirit of the open-source community. They intend to show their results to friends or colleagues or in online communities. Some mentioned specific groups on the online platform Reddit for that purpose. The answers, which we tagged as *reference*, follow a similar purpose. These include respondents who want to directly reference 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 centralized storage or backup for this kind of (configuration) 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, even though one user responded for example, that they appreciate the convenience of downloading their configuration quickly on virtual machines without any authentication, which would be necessary with a private repository. Still, in general, a private repository is most likely the best option, and easily available since GitHub nowadays allows unlimited private repositories for free.

Side-effects and 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 indicated that they were planning to make some changes to their `.dotfiles` repository. Out of those respondents, the majority were planning to go through their files or update them individually. Sill, $1.1\,\%$ of respondents is planning to make their repository private and $1\,\%$ is planning to delete their repository from GitHub altogether. Furthermore, it is interesting, that $0.7\,\%$ respondents 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 bespoke, self-written, tooling 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.

VIII. Related Work

Security and Privacy on Public Repositories. Meli et al. [30] performed the first large-scale systematic study across billions of files to measure the prevalence of secret leakage on GitHub. They attempted to identify the potential root cause of secret leakage and also touched on GitHub's metadata and developer practices of storing personal configuration files in repositories (i.e., `.dotfiles`). However, the authors only covered SSH key leakage within the ".ssh" directory. The authors noted that, even though these files accounted for only a small part of their dataset, the impact of leaked secrets from such repositories was non-trivial. Feng et al. [15] introduced a new technique for password detection, taking into account the intrinsic characteristics of textual passwords and semantic information of program elements to accurately identify hardcoded passwords from source code files written in different programming languages. They performed a large-scale systematic study to measure password leakage on GitHub and identified over 60,000 affected repositories in which most leaked passwords remained available for weeks or longer. Beyond files, Yasar [54] investigated secrets accidentally committed into repositories as part of continuous integration (CI) pipelines and automation. Finally, as mentioned in IV, approaches scanning for secrets based on regular expressions, such as ours suffer from false positives. To improve the detection accuracy of secrets found in public GitHub repositories approaches based on machine learning have been shown to be effective by Saha et al. [39] and Lounici et al. [28]

[2] "Your unofficial guide to dotfiles on GitHub." https://dotfiles.github.io/

[3] "GitHub Docs: Removing sensitive data from a repository" https://docs.github.com/en/github/authenticating-to-github/removing-sensitive-data-from-a-repository (last accessed April 2023)

Developer Studies on Security and Privacy. Bühlmann and Ghafari [10] performed a quantitative analysis of security issues reported from 2014 until 2020 across 182 different projects on GitHub and found different characteristics of the security reports and current practices among developers. Their exploratory study identified only a small group of developers actively involved in reporting and resolving security issues. Moreover, security issues progress slowly, and many have been pending for a long time. Dietrich et al. [13] conducted a combination of qualitative and quantitative study of human aspects of security misconfigurations from the operators' perspective. The authors' results indicated that most security misconfigurations have not (yet) led to security incidents, which suggests that countless undiscovered issues may be present in Internet-connected systems. They also identified interdependent facets of social (communication), structural, and institutional factors as primary facilitators of bad security posture leading to the misconfiguration phenomenon. Other than GitHub, several studies have been conducted on Stack Overflow for exploring a wide range of topics [48]. Researchers and practitioners have also investigated the practical challenges of the paradigm of developer-centered security. For example, Wijayarathna and Arachchilage [51] conducted a qualitative experimental study with 40 software developers to understand the programmer's perception of who is responsible for the end user's security of applications. Their results identified that programmers perceive it as their responsibility to ensure end users' security in application development. Further, even though programmers are not ignorant of security, they find it challenging to ensure the security of the applications they develop.

In summary, we consider the above-mentioned study by Meli et al. [30], consisting mostly of small-scale quantitative analysis of repositories containing ".ssh" files, the most closely related to our work. Their goal was to cover a wide range of different repository types that may leak credentials, while our work specifically focuses on personal configuration stored in `.dotfiles`. Compared to prior studies with repository users, we perform a dedicated large-scale analysis of 124,230 public `.dotfiles` repositories and provide comprehensive insights into the mental models of these repository owners (n=1,650) in the context of security and privacy.

IX. Conclusion

We provide comprehensive insights into the security and privacy implications of publicly sharing personal configuration files. To this end, 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 personal configuration files. 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"), as well as 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 them after taking appropriate actions.

We conclude that, in light of the popularity and widespread use of `.dotfiles` repositories, users 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.

Acknowledgements

We thank the reviewers for their comments and feedback for improving our paper. We further thank Katharina Krombholz for her valuable input concerning our study design.

This research has been funded by the Vienna Science and Technology Fund (WWTF) [10.47379/ICT19056], as well as SBA Research (SBA-K1), a COMET Centre within the framework of COMET – Competence Centers for Excellent Technologies Programme and funded by BMK, BMDW, and the federal state of Vienna. The COMET Programme is managed by FFG. We further gratefully acknowledge a career grant from the Faculty of Informatics at TU Wien.

References

[1] S. Achleitner, T. F. La Porta, P. McDaniel, S. Sugrim, S. V. Krishnamurthy, and R. Chadha. "Deceiving Network Reconnaissance Using SDN-Based Virtual Topologies". In: *IEEE Transactions on Network and Service Management* 14.4 (2017). DOI: 10.1109/TNSM.2017.2724239.

[2] A. Af Wåhlberg and L. Poom. "An Empirical Test of Nonresponse Bias in Internet Surveys". In: *Basic and Applied Social Psychology* 37.6 (2015). DOI: 10.1080/01973533.2015.1111212.

[3] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan. "The Menlo Report". In: *IEEE Security and Privacy* 10.2 (2012). DOI: 10.1109/MSP.2012.52.

[4] M. Barbulescu, A. Stratulat, V. Traista-Popescu, and E. Simion. "RSA Weak Public Keys Available on the Internet". In: *Innovative Security Solutions for Information Technology and Communications*. 2016.

[5] E. Barker. *Recommendation for Key Management: Part 1 - General*. Tech. rep. NIST Special Publication (SP) 800-57 Part 1 Revision 5. National Institute of Standards and Technology, May 2020. DOI: 10.6028/NIST.SP.800-57pt1r5.

[6] J. Bethlehem. "Selection Bias in Web Surveys". In: *International Statistical Review* 78.2 (2010). DOI: 10.1111/j.1751-5823.2010.00112.x.

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

[8] H. Borges, A. Hora, and M. T. Valente. "Understanding the Factors that Impact the Popularity of GitHub Repositories". In: *Proc. of the IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2016.

[9] H. Borges and M. Tulio Valente. "What's in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform". In: *Journal of Systems and Software* 146 (2018). DOI: 10.1016/j.jss.2018.09.016.

[10] N. Bühlmann and M. Ghafari. "How Do Developers Deal with Security Issue Reports on GitHub?" In: *Proc. of the ACM/SIGAPP Symposium on Applied Computing (SAC)*. 2022.

[11] J. Cito, P. Leitner, T. Fritz, and H. C. Gall. "The Making of Cloud Applications An Empirical Study on Software Development for the Cloud". In: *Proc. of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*. 2015.

[12] F. J. Damerau. "A Technique for Computer Detection and Correction of Spelling Errors". In: *Communications of the ACM* 7.3 (1964). DOI: 10.1145/363958.363994.

[13] C. Dietrich, K. Krombholz, K. Borgolte, and T. Fiebig. "Investigating System Operators' Perspective on Security Misconfigurations". In: *Proc. of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2018.

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

[15] R. Feng, Z. Yan, S. Peng, and Y. Zhang. "Automated Detection of Password Leakage from Public GitHub Repositories". In: *Proc. of the International Conference on Software Engineering (ICSE)*. 2022.

[16] T. Fry, T. Dey, A. Karnauch, and A. Mockus. "A Dataset and an Approach for Identity Resolution of 38 Million Author IDs Extracted from 2B Git Commits". In: *Proc. of the the International Conference on Mining Software Repositories (MSR)*. 2020.

[17] g0tmi1k. *Debian OpenSSL Predictable PRNG (CVE-2008-0166)*. URL: https://github.com/g0tmi1k/debian-ssh (visited on 02/17/2022).

[18] GitGuardian. *State of Secrets Sprawl on GitHub - 2021*. Mar. 2021. URL: https://blog.gitguardian.com/state-of-secrets-sprawl-2021/ (visited on 05/27/2021).

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

[20] G. Jungwirth, A. Saha, M. Schröder, T. Fiebig, M. Lindorfer, and J. Cito. *Dotfiles Repositories Analysis Artifact*. https://github.com/ipa-lab/dotfiles-repositories-analysis. Online Appendix. 2021.

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

[22] M. Kaur, M. van Eeten, M. Janssen, K. Borgolte, and T. Fiebig. *Human Factors in Security Research: Lessons Learned from 2008-2018*. 2021. arXiv: 2103.13287 [cs.CY]. URL: https://arxiv.org/abs/2103.13287v1.

[23] K. Krombholz, H. Hobel, M. Huber, and E. Weippl. "Advanced Social Engineering Attacks". In: *Journal of Information Security and Applications* 22 (2015). DOI: 10.1016/j.jisa.2014.09.005.

[24] O. Kuru and J. Pasek. "Improving Social Media Measurement in Surveys: Avoiding Acquiescence Bias in Facebook Research". In: *Computers in Human Behavior* 57 (2016). DOI: 10.1016/j.chb.2015.12.008.

[25] B. Lazarine, S. Samtani, M. Patton, H. Zhu, S. Ullman, B. Ampel, and H. Chen. "Identifying Vulnerable GitHub Repositories and Users in Scientific Cyberinfrastructure: An Unsupervised Graph Embedding Approach". In: *Proc. of th4e IEEE International Conference on Intelligence and Security Informatics (ISI)*. 2020.

[26] Z. Li, Y. Yu, T. Wang, S. Li, and H. Wang. "Opportunities and Challenges in Repeated Revisions to Pull-Requests: An Empirical Study". In: *Proc. of the ACM on Human-Computer Interaction* 6.CSCW2 (2022). DOI: 10.1145/3555208.

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

[28] S. Lounici, M. Rosa, C. M. Negri, S. Trabelsi, and M. Önen. "Optimizing Leak Detection in Open-source Platforms with Machine Learning Techniques". In: *Proc. of the International Conference on Information Systems Security and Privacy (ICISSP)*. 2021.

[29] N. McDonald, S. Schoenebeck, and A. Forte. "Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice". In: *Proc. of the ACM on Human-Computer Interaction* 3.CSCW (2019). DOI: 10.1145/3359174.

[30] M. Meli, M. R. McNiece, and B. Reaves. "How Bad Can It Git? Characterizing Secret Leakage in Public GitHub Repositories". In: *Proc. of the Network and Distributed System Security Symposium (NDSS)*. 2019.

[31] M. Menichinelli. "A data-driven approach for understanding Open Design. Mapping social interactions in collaborative processes on GitHub". In: *The Design Journal* 20.sup1 (2017). DOI: 10.1080/14606925.2017.1352869.

[32] D. Nakano, M. Yin, R. Sato, A. Hindle, Y. Kamei, and N. Ubayashi. *A Quantitative Study of Security Bug Fixes of GitHub Repositories*. 2020. arXiv: 2012.08053 [cs.SE]. URL: https://arxiv.org/abs/2012.08053v1.

[33] M. Ohm, H. Plate, A. Sykosch, and M. Meier. "Backstabber's Knife Collection: A Review of Open Source Software Supply Chain Attacks". In: *Proc. of the Conference Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*. 2020.

[34] G. A. A. Prana, D. Ford, A. Rastogi, D. Lo, R. Purandare, and N. Nagappan. "Including Everyone, Everywhere: Understanding Opportunities and Challenges of Geographic Gender-Inclusion in OSS". In: *IEEE Transactions on Software Engineering* 48.9 (2021). DOI: 10.1109/TSE.2021.3092813.

[35] J. J. Ray. "Reviving the Problem of Acquiescent Response Bias". In: *The Journal of Social Psychology* 121.1 (1983). DOI: 10.1080/00224545.1983.9924470.

[36] E. M. Redmiles, Y. Acar, S. Fahl, and M. L. Mazurek. *A Summary of Survey Methodology Best Practices for Security and Privacy Researchers*. Tech. rep. University of Maryland, 2017. DOI: 10.13016/M22K2W. URL: https://drum.lib.umd.edu/handle/1903/19227.

[37] Z. Rice. *Protect and discover secrets using Gitleaks*. URL: https://github.com/zricethezav/gitleaks (visited on 02/17/2022).

[38] M. O. F. Rokon, R. Islam, A. Darki, E. E. Papalexakis, and M. Faloutsos. "SourceFinder: Finding Malware Source-Code from Publicly Available Repositories in GitHub". In: *Proc. of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*. 2020.

[39] A. Saha, T. Denning, V. Srikumar, and S. K. Kasera. "Secrets in Source Code: Reducing False Positives using Machine Learning". In: *Proc. of the International Conference on COMmunication Systems NETworkS (COMSNETS)*. 2020.

[40] J. Saldana. *The Coding Manual for Qualitative Researchers*. SAGE, 2021. ISBN: 978-1-5297-5599-2.

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

[42] SchizoDuckie. *Responsible Disclosure of GitHub dotfiles ethical hack*. Dec. 2021. URL: https://twitter.com/SchizoDuckie/status/1474087696247279626?s=20.

[43] L. Singer, F. Figueira Filho, and M.-A. Storey. "Software Engineering at the Speed of Light: How Developers Stay Current Using Twitter". In: *Proc. of the International Conference on Software Engineering (ICSE)*. 2014.

[44] J. Spaulding, D. Nyang, and A. Mohaisen. "Understanding the Effectiveness of Typosquatting Techniques". In: *Proc. of the ACM/IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*. 2017.

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

[46] A. Strauss and J. M. Corbin. *Grounded Theory in Practice*. SAGE, 1997. ISBN: 978-0-7619-0748-0.

[47] The MITRE Corporation. *ATT&CK*. 2021. URL: https://attack.mitre.org/ (visited on 05/14/2021).

[48] F. Tian, P. Liang, and M. A. Babar. "How Developers Discuss Architecture Smells? An Exploratory Study on Stack Overflow". In: *Proc. of the IEEE International Conference on Software Architecture (ICSA)*. 2019.

[49] N. P. Tschacher. "Typosquatting in Programming Language Package Managers". Bachelor's Thesis. University of Hamburg, Mar. 2016. URL: https://incolumitas.com/data/thesis.pdf.

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

[51] C. Wijayarathna and N. A. Arachchilage. "Am I Responsible for End-User's Security? A Programmer's Perspective". In: *Proc. of the USENIX Symposium on Usable Privacy and Security (SOUPS)*. 2018.

[52] C. E. Wills, K. Cadwell, and W. Marrs. "Customization in a UNIX Computing Environment". In: *Proc. of the USENIX System Administration Conference (LISA)*. 1993.

[53] T. Xu, J. Zhang, P. Huang, J. Zheng, T. Sheng, D. Yuan, Y. Zhou, and S. Pasupathy. "Do Not Blame Users for Misconfigurations". In: *Proc. of the ACM Symposium on Operating Systems Principles (SOSP)*. 2013.

[54] H. Yasar. "Experiment: Sizing Exposed Credentials in GitHub Public Repositories for CI/CD". In: *Proc. of the IEEE Cybersecurity Development Conference (SecDev)*. 2018.

[55] S. Yilek, E. Rescorla, H. Shacham, B. Enright, and S. Savage. "When Private Keys Are Public: Results from the 2008 Debian OpenSSL Vulnerability". In: *Proc. of the ACM SIGCOMM Conference on Internet Measurement (IMC)*. 2009.

[56] Y. Yu, G. Yin, H. Wang, and T. Wang. "Exploring the Patterns of Social Behavior in GitHub". In: *Proc. of the International Workshop on Crowd-Based Software Development Methods and Technologies (CrowdSoft)*. 2014.

Appendix

A. List of Regular Expressions

We used the following list of regular expressions to identify relevant secrets and information in `.dotfiles` repositories:

```
description = ‘Email Simple’
email = \b[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]
+\.[a-zA-Z0-9-.]+\b

description = ‘Firefox Profile’
path =mozilla/firefox.*(logins\.json|
cookies\.sqlite|places\.sqlite)

description = ‘Files with credentials’
file = (?i)(id_rsa|passwd|id_rsa.pub|
pgpass|pem|key|shadow)

description = ‘Thunderbird Profile’
path = ($|/)\.?thunderbird/

description = ‘Crypto Wallet’
file = wallet\.dat

description = ‘Chrome Profile’
path = config.*/(google-chrome|chromium)/
```

B. Survey Questionnaire

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 the BIG (Business Informatics group) and S&P (Security and Privacy group) at TU Wien, Austria. 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 gerhard.jungwirth@tuwien.ac.at.

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)?

C. Recruitment Mail

We sent the following recruitment mail to the repository owners. Depending on the issues we identified, we either listed all the issues found or to stated, "No leaks were found in your repository," in case of no issues.

"Hello *username*,

We are a research team at TU Wien. 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):

- 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 somewhere, 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)"