2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR) | 979-8-3503-1184-6/23/$31.00 ©2023 IEEE | DOI: 10.1109/MSR59073.2023.00053

SecretBench: A Dataset of Software Secrets

Setu Kumar Basak*, Lorenzo Neil†, Bradley Reaves‡ and Laurie Williams§
North Carolina State University, USA
Email: *sbasak4@ncsu.edu, †lcneil@ncsu.edu, ‡bgreaves@ncsu.edu, §lawilli3@ncsu.edu

Abstract—**According to GitGuardian's monitoring of public GitHub repositories, the exposure of secrets (API keys and other credentials) increased two-fold in 2021 compared to 2020, totaling more than six million secrets. However, no benchmark dataset is publicly available for researchers and tool developers to evaluate secret detection tools that produce many false positive warnings. *The goal of our paper is to aid researchers and tool developers in evaluating and improving secret detection tools by curating a benchmark dataset of secrets through a systematic collection of secrets from open-source repositories.* We present a labeled dataset of source codes containing 97,479 secrets (of which 15,084 are true secrets) of various secret types extracted from 818 public GitHub repositories. The dataset covers 49 programming languages and 311 file types.**

I. Introduction

GitGuardian reported in March 2022 that the number of secrets leaked on GitHub repositories doubled in 2021 compared to 2020, totaling more than six million secrets [1]. Often, a software program uses third-party services, including payment systems, location services, and social media integration. Software developers need secrets (API keys, access tokens, and private keys) to authenticate these third-party services as part of system integration. However, developers may expose these secrets in plain text in the version control systems (VCS) or the application packages [2], [3]. Although the problem with checked-in secrets is well known, the secret leakage incidents continued. On September 2022, Uber confirmed an organization-wide cybersecurity breach because of having hard-coded secrets in a PowerShell script [4]. The attackers got the administrator access and compromised Uber's AWS, GCP, Google Drive, and Slack workspaces.

To avoid exposing secrets in VCS, several open-source and proprietary secret detection tools [5], such as TruffleHog [6] and Microsoft CredScan [7], are available. However, these tools have been shown to produce false positive warnings [8]. In previous studies [9], [10], researchers have worked on reducing false positives. However, their curated datasets are not large and varied and are unavailable for future research and evaluation purposes. In addition, developers face challenges in choosing one tool out of many, and no publicly-available dataset is available for comparing the effectiveness of the tools.

The goal of our paper is to aid researchers and tool developers in evaluating and improving secret detection tools by curating a benchmark dataset of secrets through a systematic collection of secrets from open-source repositories.

We present SecretBench, a labeled dataset of source codes consisting of 97,479 secrets extracted from 818 public GitHub repositories using two secret detection tools. We manually inspected each secret and labeled 15,084 secrets as true secrets. The dataset encompasses 49 programming languages and 311 file types. The dataset is hosted in Google BigQuery [11] and Cloud Storage [12] and designed to be amenable to expansion by the community. Our dataset will aid in expediting the research to evaluate and improve secret detection tools.

II. Data Extraction

We provide our eight-step process for data collection of SecretBench as follows:

Step 1: Open Source Software Repository Platform Selection: We choose GitHub [13] to select candidate repositories containing secrets for our study. GitHub is the most popular platform for hosting open-source software development projects [14]. As of December 2022, GitHub has over 94 million developers and more than 330 million repositories [14], including at least 36 million public repositories [15].

Step 2: Build Regular Expression (Regex) Pattern Set: We build a regex pattern set for different types of secrets to identify the candidate repositories containing secrets for our study. For example, the regex pattern for a Slack token is "`(xoxb|xoxp|xapp|xoxa|xoxr)-[0-9]10,13[a-zA-Z0-9]*`". TruffleHog [6], a popular open-source secret-scanning tool, has a package of secret detectors [16]. We extracted 751 regex patterns from the source code of the detector package and included those in our pattern set. In addition, we included 10 regex patterns from Meli et al. [2] to find the presence of secrets in GitHub repositories that are not present in the TruffleHog detector package. In total, we used 761 regex patterns in our pattern set, which is available online [17].

Step 3: Identify Candidate Software Repositories: To identify the candidate software repositories, we used the Google BigQuery Public Dataset of GitHub [11] (Dataset ID: *bigquery-public-data.github_repos*), which was released in 2016 by Google in collaboration with GitHub. The source code of over 2 billion files from more than 2.9 million open-source licensed repositories can be accessed with SQL queries [11]. We used the most recent snapshot available at the start of this project (September 20, 2022). We wrote an SQL script with all the 761 regex patterns to search for secrets in the source code files and executed the script in Google BigQuery. The SQL script took almost 22 hours to complete, as every file is checked with all the regex patterns. The returned result is a table of two columns: "repo_name" and "matches". The "repo_name" column represents the repository name, and the "matches" column represents the list of regex patterns matched

2574-3864/23/$31.00 ©2023 IEEE
DOI 10.1109/MSR59073.2023.00053

with the specific repository. In total, we have found 2,234,618 repositories with at least one regex pattern match.

Step 4: Apply Selection Criteria on Candidate Repositories: As suggested by prior research [18], GitHub repositories need to be curated by removing inactive, beginner, and tech-demo projects. To curate the repositories collected in Step 3, we collected fork information, contributor counts, and commit counts using the GitHub Rest API [19]. We applied the following selection criteria to curate the collected repositories. The number in parenthesis with the criteria name indicates the number of filtered repositories after applying that specific criteria.

- **Availability (2,013,913):** The repository is available to download.
- **Uniqueness (1,735,864):** The repository is not a forked repository. This criteria is applied to avoid near duplicates of the same repository.
- **Collaboration (889,984):** The repository contributor count must be at or above the dataset median of 2. This criteria is applied to avoid personal or hobby projects.
- **Development History (622,719):** The repository commit count must be at or above the dataset median of 20 commits.
- **Recent Activity (93,958):** The repository must have at least one commit in the last one year. This criteria is applied to avoid inactive projects.

In addition, we observed some repositories with different "repo_name" fields point to the same repository. For example, repositories "Jasig/cas" and "apereo/cas" are the same repository though having different repository names in the dataset. This duplication happened because the repository owner changed the repository name at some point, but the Google BigQuery dataset kept both names. However, GitHub stores the actual repository name of the duplicate repository. We collected the actual repository name of each repository using the GitHub Rest API and filtered the duplicate repositories. After all selection criteria, we passed 89,070 unique repositories to Step 5.

Step 5: Find Multiset-Multicover Repositories of Regex Patterns: In this step, we further select repositories so that we get a sample of multiple secrets for each secret type while minimizing the overall repository count of the dataset. In later steps, we manually determine if identified secrets were actually secret or not. However, identifying and manually labeling secrets from the 89,070 repositories remaining in Step 4 is impractical. Our goal of identifying the smallest selection of repositories that altogether include a specified count of each identified secret pattern is actually an instance of the *multiset-multicover* problem, so we applied the multiset-multicover algorithm described in Algorithm 1. This algorithm is an extension of the Minimum Set Cover algorithm [20] to select a minimal set of repositories covering all the regex patterns with a certain number of repositories for each pattern.

Before applying the multiset-multicover algorithm, we observe that 390 out of 761 regex patterns found no match in any repository. The median regex pattern matched 10 repositories, with 186 regex patterns matching 10 or more repositories. We term these patterns "upper tail" regex patterns. An additional 120 regex patterns matched between 1 and 9 repositories; we will refer to these as "lower tail" regex patterns. The median lower tail regex pattern matched 2 repositories.

For a comprehensive dataset, we seek a balance between examples of common and uncommon secret types, so we applied the multiset-multicover algorithm in two phases. In Phase 1, we ran the multiset-multicover algorithm for the 186 upper tail regex patterns to find a set of repositories where each regex pattern matches at least 10 repositories. We identified 649 repositories among the upper tail regex patterns. For Phase 2, we ran the multiset-multicover algorithm for the 120 lower tail regex patterns to find a set of repositories where each regex pattern should match at least 2 repositories and identified 190 repositories. Then, we merged the repositories of Phase 1 with Phase 2 and removed duplicate repositories. Altogether, we identified 818 repositories for SecretBench to collect candidate secrets.

Algorithm 1 Multiset-Multicover Algorithm

```
Require: $PatternsToCover, U$
Require: $InstanceSize, K$
$R_a \leftarrow ReadAllRepos()$
$CoveredRepos, C_r \leftarrow \emptyset$
$CoveredPatterns, C_p \leftarrow \emptyset$
while $C_p \neq U$ do
    $M \leftarrow FindRepoWithMostUncoveredPatterns(R_a, C_p, U)$
    $C_p \leftarrow C_p \cup FindMatchedPatternsForRepo(M, R_a)$
    $C_r \leftarrow C_r \cup M$
end while
$R_{cc} \leftarrow FindRepoCountPerPatternInInitialCover(C_r, R_a, U)$
$U_p \leftarrow FindPatternsLessThanKInstance(R_{cc})$
while $len(U_p) \neq 0$ do
    $M \leftarrow FindRepoWithMostUncoveredPatterns(R_a, C_p, U_p)$
    $R_p \leftarrow FindMatchedPatternsForRepo(M, R_a)$
    $C_p \leftarrow C_p \cup R_p$
    for $e\ in\ R_p$ do
        $R_{cc}[e] \leftarrow R_{cc}[e] + 1$
    end for
    $U_p \leftarrow FindPatternsLessThanKRepoInstance(R_{cc})$
    $R_a \leftarrow RemoveSelectedRepoFromList(M, R_a)$
    $C_r \leftarrow C_r \cup M$
end while
$C_r \leftarrow RemoveDuplicateRepos(C_r)$
return $C_r$
```

Step 6: Find Candidate Secrets: We wrote a Python program to clone the repositories. We used GitPython [21] to download all the branches of a repository and saved the files into a Google Cloud VM Instance [22] (OS: Ubuntu 18.04 LTS, RAM: 16 GB, Persistent Disk: 500 GB). Next, we ran two secret detection tools, TruffleHog [6] and Gitleaks [23], to identify candidate secrets from the repositories. Both tools are widely used for secret detection and can identify secrets buried in the repository's history and logs. We used these tools since manually inspecting each file of a repository to find secrets is infeasible and would be error-prone. The tools provide a JSON output for each repository. The JSON output contains the candidate secrets with additional metadata such as the commit id, commit date, committer email, file path, start line, end line, start column, and end column of the file where secrets are matched. Next, we wrote another Python program to read each report generated by the tools and extract the candidate secrets along with the metadata. Altogether, we identified 97,479 candidate secrets present in different commits of 818 repositories, of which 27,336 secrets are unique.

Step 7: Label Candidate Secrets: The first and second authors manually inspected each candidate secret independently using the metadata collected in Step 6. A candidate secret is labeled as “True” if the secret is a true secret, otherwise labeled as “False”. We observed the agreement of the labeling of secrets with a Cohen’s Kappa [24] score of 0.86 between two raters, which indicates a “near perfect agreement” according to Landis and Koch’s interpretation [25]. The disagreements were resolved after a discussion between the two raters. In our dataset, we identified 15,084 true secrets, of which 4,014 secrets are unique.

Step 8: Developer Survey: We conducted a developer survey to evaluate whether the committer of the secrets agrees with our label. First, we selected unique secrets committed between 2021 and 2022 to avoid recall bias [26] from the developers and identified 7,617 secrets. Since GitHub allows the developers to use a noreply email address (`user-name@users.noreply.github.com`) as the commit email address [27], we filtered those secrets and identified 2,115 secrets. Then, we selected 200 secrets (randomly selected to avoid selection bias [28]) and emailed the developers to know if they agreed with our labeling of the secret and the reason they disagreed. In the email, we provided the repository name, commit id with the commit GitHub link, file path, start line, end line, and a screenshot of the code where the secret is found. We received 56 responses, a 28.0% response rate. Altogether, 44 (78.6%) respondents fully agreed with our label, while 6 (10.7%) respondents disagreed. The remaining 6 (10.7%) respondents were not sure.

III. Data Description

In this section, we provide brief details of our dataset.

A. *Curated and Derived Fields:*

We collected the metadata related to the secret such as repository name, commit id, commit date, committer email, file path, start line and end line. To further enrich the dataset, we have augmented the mined data with additional features that are computed or derived from the source code files and secrets. Example of computed and derived fields are “file_type”, “is_template”, “in_url”, “entropy”, “character_set” and “has_words”. An overview of our SecretBench dataset is presented in Table I.

B. *Data Characteristics*

Our SecretBench dataset is diverse in terms of different project characteristics. The dataset consists of 97,479 secrets in 818 repositories, and some repositories use multiple programming languages. For example, the repository “paradite/hn-ratio” [30] consists of two programming languages: JavaScript and Shell. Altogether, our dataset repositories used 49 programming languages. The top 5 programming languages based on the number of repositories are Shell (459), JavaScript (414), Python (312), Java (180), and Ruby (172). The number in parenthesis denotes the number of repositories containing the specific language. In addition, our dataset consists of secrets present in 311 file types. The top 5 file types based on the number of secrets in those files are js (10,412), nix (8,623), json (8,132), txt (7,737), and xml (6,429). Besides, the top 5 file types based on the number of true secrets are txt (2,935), toml (1,985), js (1,583), html (1,337), and pem (813). The number in parenthesis denotes the number of secrets in the specific file type.

TABLE I: Overview of the SecretBench Dataset

Field Name	Description	Data Type
id	Unique identifier of the secret.	Integer
secret	Candidate secret string.	String
repo_name	Name of the repository.	String
domain	Domain of the repository such as GitHub.	String
commit_id	Commit hash where the secret is added.	String
file_path	File path where the secret is included.	String
file_type	Type of the file such as .py and .config.	String
start_line	Start line no. where the secret is present.	Integer
end_line	End line no. where the secret is present.	Integer
start_column	Start index of the secret in the start line.	Integer
end_column	End index of the secret in the end line.	Integer
committer_email	Email address of the committer.	String
commit_date	The timestamp of the commit.	TimeStamp
label	The ground truth label of the secret.	Boolean
is_template	Flag to indicate if the secret is a placeholder such as “MY_PASSWORD".	Boolean
in_url	Flag to indicate if the secret is part of URL such as “http://user:pwd@site.com".	Boolean
entropy	Shannon entropy of the secret.	Float
character_set	Characters used in the secret (NumberOnly, CharOnly, Any).	String
has_words	Flag to indicate if any common English word [29] of at least length of 4 is present within the secret.	Boolean
length	Length of the secret.	Integer
is_multiline	Flag to indicate if the secret is present in multiple lines.	Boolean
category	The category of the secret.	String
file_identifier	Unique identifier of the file to check the secret from local system.	String
repo_identifier	Unique identifier of the repository to check the secret from local system.	String

The secrets in our dataset are categorized into eight categories and presented in Table II, sorted based on the number of true secrets. More details of our dataset is presented in our GitHub repository [31].

C. *Data Storage*

Our dataset is stored as relation structured data in Google BigQuery (Dataset ID: *dev-range-332204.secretbench.secrets*). Users can run SQL queries to access and expand the dataset. In addition, we stored the downloaded 818 repositories and the secret-containing individual source code files in Google Cloud Storage. When downloaded into the local system, the “repo_identifier” and “file_identifier” mentioned in Table I can

TABLE II: The categories of secrets in SecretBench

Category	True Secrets	Total Secrets
Private Key	5,789	8,584
API Key and Secret	4,529	5,162
Authentication Key and Token	3,569	5,833
Other	524	66,690
Generic Secret	334	439
Database and Server URL	162	9,970
Password	150	705
Username	27	96

be used to locate the repository and specific source code file related to the secret, respectively.

Since our dataset is sensitive, Google BigQuery and Cloud Storage enable us to give access to the dataset to only selected groups, such as fellow researchers and tool developers. To get access to our dataset, researchers and tool developers need to contact us through email.

IV. Originality of SecretBench

Previous studies [2], [9], [10] have extracted secrets from the GitHub repositories, but none made their dataset public for future research purposes. Saha et al. [9] created a labeled dataset of 5000 secrets (700 true secrets) from 300 GitHub repositories using 32 regex patterns. With the dataset, they applied machine learning algorithms to distinguish true secrets. However, the repositories matched by regex patterns are not filtered for demo and inactive projects, and no information is provided on the files and languages covered. Sinha et al. [10] created a dataset of 84 GitHub repositories and identified pattern-based search and heuristics-driven filtering approaches to reduce the false positive detection of secrets. However, their dataset is small and contains only AWS credentials.

On the other hand, our dataset presented herein is large and diverse. We applied 761 regex patterns of different types of secrets and selected 818 GitHub repositories encompassing 49 programming languages. Our dataset consists of 97,479 labeled secrets, including 15,084 true secrets present in 311 different file types. We also provided different features related to the secret, such as whether the secret is a template or present in a URL. In addition, we will make our dataset available for future researchers and tool developers.

V. Research Opportunities

To prevent exposing secrets in VCS, there are several open-source and proprietary secret detection tools [5]. However, these tools are known to generate false positive warnings [8], [9]. Researchers and tool developers can identify different rules and patterns from false positive secrets to reduce false positive warnings. However, mining data from open-source and building ground-truth datasets is challenging and time-consuming. In this case, our SecretBench dataset can be used to circumvent the challenge and speed up the research and tool evaluation on reducing false positives. In addition, since several secret detection tools exist, developers face difficulty choosing one tool out of many. Future research is needed to aid developers in making informed choices about using different secret detection tools through an analysis of the effectiveness of the tools. In this case, our SecretBench dataset can act as a benchmark for comparing the effectiveness of the secret detection tools.

Dataset Enhancement: Our dataset can be further improved by including repositories from other VCS services such as GitLab and Bitbucket. In addition, we can add more features regarding secrets to help in secret detection automatically using machine learning algorithms. Example features include whether the secrets have parentheses (possible function call), begin with a $ sign (possible variable), and have context words such as “dummy” and “fake” in the surrounding code of the secret. We released these additional features online [32].

VI. Ethics and Data Protection

Since our dataset contains sensitive information such as true secrets and the committer’s email addresses, we will distribute our dataset selectively. Researchers and tool developers who want to use our dataset will sign a data protection agreement with us to avoid any unethical use. After that, we will give access to our dataset from Google BigQuery and Cloud Storage using their email addresses. In addition, at no point we did not attempt to use the secrets to verify the validity of the secrets. Instead, we labeled the secrets only by inspecting the secrets and the source code context of the secrets.

To validate our labeling, we only contacted the developers who committed the secrets. We did not reveal the identity of the developers to any managers or higher officials where they work. In addition, we are notifying every developer in our dataset to remove the secrets from their VCS.

VII. Threats to Validity

In this section, we briefly discuss the limitations of our paper. **VCS Selection:** We did not consider other VCS services such as GitLab [33] and Bitbucket [34]. In the future, we plan to expand our dataset by including repositories of other VCS services. **Manual Analysis Bias:** The labeling of the secrets in our dataset is susceptible to bias. To mitigate the bias, a second rater labeled the secrets independently, and we resolved the disagreements. **Recall Bias:** For the developer survey, though we have selected secrets that are committed in 2021 and 2022, the responses could have recall bias. We provided the developers with a screenshot of the secret-containing source code and additional metadata to mitigate the bias.

VIII. Conclusion

We provide the SecretBench dataset consisting of 97,479 labeled secrets extracted from 818 GitHub repositories encompassing 49 programming languages and 311 file types. Our dataset will aid in evaluating and improving secret detection tools, thus preventing secret leakage in VCS and application packages. By adding new projects and features, we aim to expand our dataset. We invite the research community to join our effort to expand and enrich the dataset to create novel software secret management research opportunities.

Acknowledgement

This work was supported by National Science Foundation 2055554 grant and the Google Cloud Research Credits program GCP19980904 award. The authors would also like to thank the Realsearch research group for their valuable input on this paper.

References

[1] T. Segura, "The State of Secrets Sprawl 2022," https://blog.gitguardian.com/the-state-of-secrets-sprawl-2022, [Online; accessed Jan 8, 2023].

[2] M. Meli, M. R. McNiece, and B. Reaves, "How bad can it git? characterizing secret leakage in public github repositories," *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

[3] S. Nichols, "Popular mobile apps leaking aws keys, exposing user data," https://www.techtarget.com/searchsecurity/news/252500361/Popular-mobile-apps-leaking-AWS-keys-exposing-user-data, [Online; accessed Jan 2, 2023].

[4] M. Jackson, "Uber Breach 2022 – Everything You Need to Know," https://blog.gitguardian.com/uber-breach-2022, [Online; accessed Jan 4, 2023].

[5] "Nine DevSecOps secret scanning tools to keep the bad guys at bay," https://www.cybersecasia.net/tips/nine-devsecops-scanning-tools-to-keep-the-bad-guys-at-bay, [Online; accessed Jan 7, 2023].

[6] "TruffleHog," https://trufflesecurity.com/trufflehog, [Online; accessed Jan 4, 2023].

[7] "Getting started with Credential Scanner (CredScan)," https://secdevtools.azurewebsites.net/helpcredscan.html, [Online; accessed Jan 2, 2023].

[8] M. R. Rahman, N. Imtiaz, M.-A. Storey, and L. Williams, "Why secret detection tools are not enough: It's not just about false positives-an industrial case study," *Empirical Software Engineering*, vol. 27, no. 3, pp. 1–29, 2022.

[9] A. Saha, T. Denning, V. Srikumar, and S. K. Kasera, "Secrets in source code: Reducing false positives using machine learning," in *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 2020, pp. 168–175.

[10] V. S. Sinha, D. Saha, P. Dhoolia, R. Padhye, and S. Mani, "Detecting and mitigating secret-key leaks in source code repositories," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 396–400.

[11] "GitHub on BigQuery: Analyze all the open source code," https://cloud.google.com/blog/topics/public-datasets/github-on-bigquery-analyze-all-the-open-source-code, [Online; accessed Jan 5, 2023].

[12] "Google Cloud Storage," https://cloud.google.com/storage, [Online; accessed Dec 24, 2022].

[13] "GitHub," https://github.com, [Online; accessed Jan 3, 2023].

[14] "About GitHub," https://github.com/about, [Online; accessed Jan 3, 2023].

[15] "GitHub Public Repositories," https://github.com/search?q=is:public, [Online; accessed Jan 3, 2023].

[16] "TruffleHog," https://github.com/trufflesecurity/trufflehog/tree/main/pkg/detectors, [Online; accessed Jan 4, 2023].

[17] "SecretBench Regular Expressions," https://doi.org/10.5281/zenodo.7555981, [Online; accessed Jan 22, 2023].

[18] E. Kalliamvakou, G. Gousios, K. Blincoe, L. Singer, D. M. German, and D. Damian, "The promises and perils of mining github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 92–101. [Online]. Available: https://doi.org/10.1145/2597073.2597074

[19] "Getting started with the REST API," https://docs.github.com/en/rest/guides/getting-started-with-the-rest-api?apiVersion=2022-11-28, [Online; accessed Jan 2, 2023].

[20] N. E. Young, *Greedy Set-Cover Algorithms*. Boston, MA: Springer US, 2008, pp. 379–381. [Online]. Available: https://doi.org/10.1007/978-0-387-30162-4_175

[21] "GitPython," https://github.com/gitpython-developers/GitPython, [Online; accessed Dec 24, 2022].

[22] "Google Cloud Compute Engine," https://cloud.google.com/compute, [Online; accessed Dec 26, 2022].

[23] "Gitleaks," https://gitleaks.io, [Online; accessed Dec 26, 2022].

[24] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960. [Online]. Available: https://doi.org/10.1177/001316446002000104

[25] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977. [Online]. Available: http://www.jstor.org/stable/2529310

[26] Spencer EA, Brassey J, Mahtani K, "Recall bias," https://www.catalogueofbiases.org/biases/recall-bias, [Online; accessed Jan 3, 2023].

[27] "Setting your commit email address," https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-personal-account-on-github/managing-email-preferences/setting-your-commit-email-address, [Online; accessed Dec 22, 2022].

[28] Nunan D, Bankhead C, Aronson JK, "Selection bias," https://catalogofbias.org/biases/selection-bias, [Online; accessed Jan 3, 2023].

[29] "Google Common English Words," https://github.com/first20hours/google-10000-english, [Online; accessed Dec 21, 2022].

[30] "paradite/hn-ratio," https://github.com/paradite/hn-ratio, [Online; accessed Jan 17, 2023].

[31] "SecretBench GitHub Repository," https://github.com/setu1421/SecretBench, [Online; accessed Jan 22, 2023].

[32] "SecretBench Additional Features," https://doi.org/10.5281/zenodo.7555981, [Online; accessed Jan 22, 2023].

[33] "GitLab," https://about.gitlab.com, [Online; accessed Jan 8, 2023].

[34] "Bitbucket," https://bitbucket.org, [Online; accessed Jan 8, 2023].