SCTE CABLE-TEC EXPO FAST FORWARD 2021
VIRTUAL EXPERIENCE
OCTOBER 11-14

2021 Fall Technical Forum
SCTE • NCTA • CABLELABS

xGitGuard: ML-based Secret Scanner for GitHub

A Technical Paper prepared for SCTE by

Bahman Rashidi
Senior Security Architect
Comcast Cable
Philadelphia PA
bahman_rashidi@comcast.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

List of Tables

Title **Page Number**

1. Abstract

Misused leaked secrets on code sharing platforms such as GitHub (GH) have caused some of the data breaches of our time. Unfortunately, this kind of credential leak is quite common across the code sharing platforms. Developers and code contributors are required in many cases by organization's security policies to comply with security practices and remove sensitive information before they push their code to GitHub. However, sometimes inadvertently developers neglect to remove sensitive information, such as API tokens and user account credentials, from their code prior to posting it. Malicious attackers crawl through GitHub, hoping to find these secrets and thus grab foothold into an organization's territory. Companies have limited ability to address this risk as given the scale of GitHub it is difficult if not impossible to find leaked secrets before malicious attackers. Some companies leverage bug bounty programs as a way to incentivize third party agents to manually look for and report these secrets through responsible disclosure. Unsurprisingly, this process can create unnecessary exposure. Consequently, we at Comcast Cybersecurity Research designed and developed "xGitGuard,"a Machine Learning (ML)-based tool that uses advanced Natural Language Processing (NLP) to detect organizational secrets and user credentials at scale and with appropriate velocity in GitHub repositories. This paper begins with a description of the problem statement. Next, we discuss the design of xGitGuard and how it improves upon current solutions, and the solution space. Finally, we provide details about how xGitGuard can be deployed in different scenarios.

2. Introduction

GitHub is the biggest open-source community with more than 200 million repositories (and of those, 30+ million are public) and over 65 million users [1] [2]. Users on GitHub publish the code of their projects, collaboratively develop software, and use the code from the platform in development. Given the open nature of GitHub, there is an opportunity to publicly disclose otherwise confidential data. [3] [4]. For example, a project may disclose login credentials for remote servers and service accounts or API tokens if such information is embedded in code distributed on the platform.

Although the risk of disclosure on GitHub and Stackoverflow is known, it is not known the extent of confidential information disclosed, and how efficiently that information is identified by attackers.

In this paper, we describe xGitGuard, an ML-based tool that detects exposed organizational secrets on GH both the public and enterprise versions. xGitGuard is designed and developed with the goal of addressing the existing challenges associated with classic regex scanning approaches (solutions relying on finding regular expressions). xGitGuard takes advantage of new text processing algorithms that can find secrets within files with high level accuracy. xGitGuard scans the entire GH for secrets efficiently using an agile scanning search approach. The agility of xGitGuard helps incident response teams to take proper actions in timely manner. In next sections, we detail how it works and can be deployed.

The rest of this paper is organized as follows: we introduce an overview of xGitGuard in Section 3, alongside a workflow overview. We then further detail the components of the xGitGuard and how they work with each other. We then discuss the related work in Section 4. We finally discuss different deployment scenarios for xGitGuard in Section 5.

3. xGitGuard

xGitGuard is an ML-based tool that detects organizational secrets, including API tokens/keys and user credentials exposed on both public and enterprise GitHub. xGitGuard has two separate models: one for detecting credentials, such as password-like secrets, and one for detecting API tokens and keys. Table 1

shows examples of such secrets. It is designed to be both scalable and accurate in scanning and detecting secrets. The rest of this section details the architecture of xGitGuard and all of its components.

Table 1 – Examples of secrets for each category

Credentials	Tokens & Keys
Username & passwords	API tokens (AWS, Azure, Slack, etc.)
Server credentials	Encryption keys
Account credentials	Session tokens
DB access credentials	Session IDs

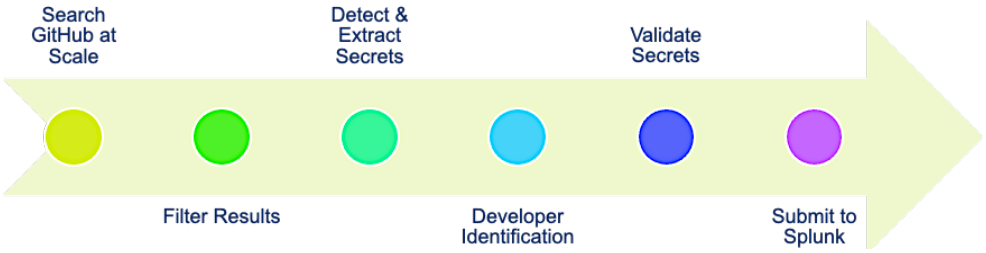

Figure 1 – xGitGuard workflow overview

3.1. Architecture

xGitGuard scans GH repos for documents potentially related to an organization and containing secrets. It also processes candidate documents for secrets, identifies developers using commit history, validates detected secrets using an ML component, and then calculates a confidence level for every detection to submit to a dashboarding tool. Figure 1 shows an overview of xGitGuard's overall workflow. The implementation of xGitGuard includes six main components. Figure 2 and Figure 3 show overview architectures of xGitGuard's workflow for credential and token detection models:

Search: xGitGuard has a unique approach to searching GitHub: it uses two types of keywords to craft GitHub queries, each for a different purpose. i) Primary keywords (PKEY): helps to search for documents that are related to the organization. ii) Secondary keys (SKEY): are then used to target documents that potentially contain secrets. xGitGuard uses two different lists of secondary keywords for credential and token detection. The two lists are deliberately different, as each will detect different types of secrets (credentials and tokens). However, the primary keywords are the same between both. With this unique approach, we scan the entire GitHub but only target documents that are relevant and sensitive.

Query Engine: This engine implements a multi-processing system that runs multiple queries in parallel to reduce the time to detect a secret. This helps with scalability. xGitGuard also maintains a hash list of documents that have been processed in the past and skips them in the future scans if they show up in query results. This way a file will be processed once, which reduces the processing time in total.

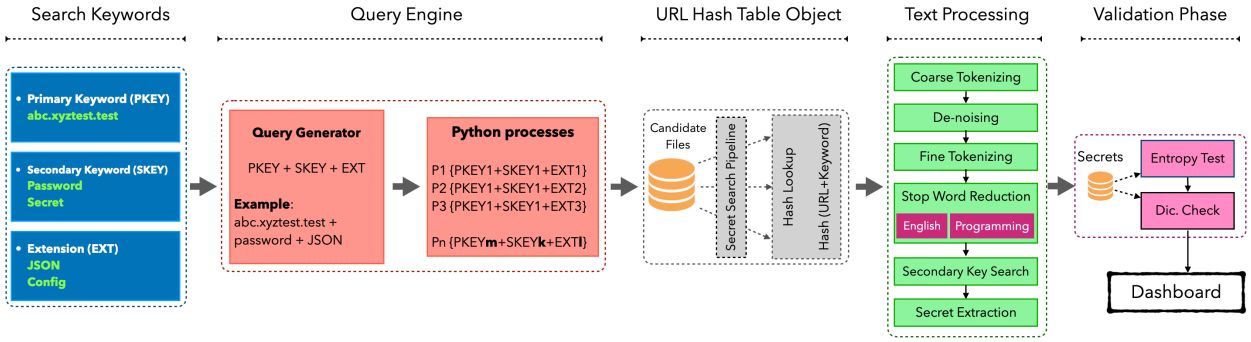

Figure 2 – Credential detection architectural overview

AI Model: This is the core of the xGitGuard, that processes the documents for secrets. This component is responsible for detecting the secrets within the documents. The core model of xGitGuard uses NLP and other in-house developed text processing algorithms to identify secrets. The model begins by breaking down documents to smaller tokens, then it removes the noise and stop words (meaning common English dictionary words, such as "the," "a" and "an," that are mutual terms/words among different languages, etc.), extracts the secrets and extra metadata (e.g., masked secret, line of code, etc.) around the detection. As Figure 3 shows, the only difference between credential and token detection models is that token detection model relies on using regexes (regular expressions) after de-noising the document.

Scoring: After detecting a secret in a document, this component calculates a confidence score for it. The higher the score, the more accurate the detection. This score is calculated based on several factors, such as the entropy of secret keys, n-gram similarity to English words, etc.

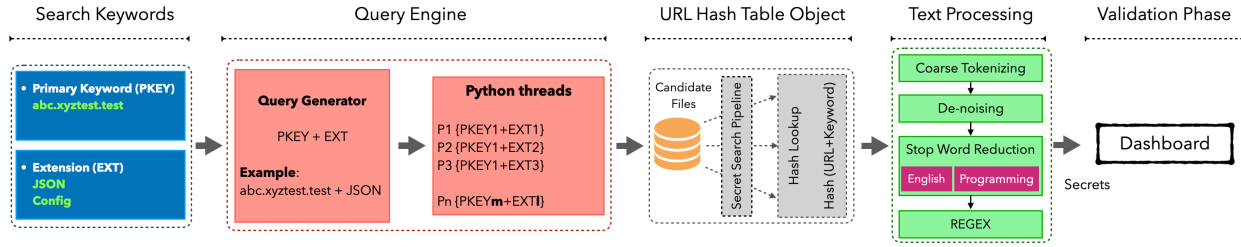

Figure 3 – Key & Token detection architectural overview

Validation Model: This ML-based model is responsible for validating detected secrets. The input to the model is a secret, and the output is whether the input is an actual secret or not. The model is built using a number of features, with the focus on the secret itself and the line of code where the secret was found. This model has an accuracy of above 90% in recognizing secrets from non-secrets text. The accuracy of the validator depends on how well we train the model and how comprehensive our training data is.

User Identification: xGitGuard is able to identify the developer who posted the code on GitHub, using commit logs related to the repos. This component extracts the email address and full name related to the developer.

3.1. Development and Deployment

The entire xGitGuard code base is developed in Python. This includes our algorithms, APIs, scanning and pre-processing documents, as well as submitting detections to dashboards. In order for xGitGuard to

perform better, all the text processing models and algorithms are persisted in the form of pickle files (serialized object models). This helps to avoid retraining our models every time a new cycle of xGitGuard starts. It also enhances agility and eases the process of deploying the final model. The modularity of pickle files helps to deploy the final model on many platforms.

4. Related Technologies

The problem of detecting leaked secrets within GH and other similar code-sharing platforms (e.g., Stackoverflow) is not a new problem in the information security community. In the past years, a number of tools from both the open source and commercial sectors tried to address such problems. However, existing solutions have some challenges, in particular scalability and accuracy.

There are three main limitations related to the existing tools. The first is that most of these tools, and especially the open source versions, exclusively focus on a single use case (they either focus on API tokens or passwords, not both). The second limitation is that almost all the solutions discussed here rely on pattern detection (regular expressions). This limits the types of secrets that are covered and only detects certain types of secrets. Finally, these approaches require users to point the tools to specific repos that they want scanned, because they lack scalability to scan the entire public or enterprise GH.

4.1. Truffle Hog

Truffle Hog is a free and open source tool that can help developers check for any hard-coded secrets. Truffle Hog is designed to scan for secrets through repositories (users need to point the tool to specific repos in order to scan them) and the entire commit history. The tool specifically searches for each diff (differences between file content) from each commit and evaluates them for secrets [5]. The Truffle Hog's core model relies on detecting high-entropy strings (average number of bits per symbol needed to encode a string) that could represent secrets, whether API tokens and keys or other types of credentials [6].

4.2. Nightfall for GitHub

Nightfall for GitHub is a commercial variant of similar tools that uses ML to detect secrets exposed on GitHub repos. Similar to Truffle Hog, this solution only works on manually-selected repos. Unlike Truffle Hog, this tool can be integrated with GH accounts in the form of an app and can automatically detect secrets as a user pushes code to repositories. This tool is an extension of Truffle Hog in terms of the approach to secret validation [7]. The main focus of Nightfall is to detect API keys and tokens and omits passwords and other credentials as a category of leak.

4.3. Gitguardian

Similar to Nightfall, Gitguardian is a commercial tool with a limited amount of publicly-available information. It describes the tool as ML-based and able to identify more than 200 types of API tokens. However, similar to Nightfall, Gitguardian exclusively targets API tokens and keys by relying on regular expression classifiers. Alongside the commercial variant that it offers, Gitguardian also provides a free service to scan one's repos on GitHub [8] [9].

4.4. EarlyBird

EarlyBird, developed by American Express, is another sensitive information detection tool relying on pattern detection (regular expression). EarlyBird exclusively scans repositories for clear text password violation, personally identifiable information (PII), sensitive file names and outdated cryptography

methods, etc. It functions on committed code (GH repos) and local files. Similar to Nightfall, it can also work as a pre-commit check. Besides the regular expressions that are used for secret detection, EarlyBird also uses entropy calculation for password detection [10].

5. Deployment

xGitGuard is a standalone and internally-developed and used application. Depending on the use case, it can be internally deployed and utilized in different ways. In this section, we discuss some of the ways that xGitGuard could be deployed.

5.1. Continuous Scanning

One way to deploy xGitGuard is to run it as a stand-alone application continuously running and scanning GitHub repos. This way, xGitGuard will continuously query GH in a cycle, receive documents and process them for secrets. This process continues until stopped manually. Using this approach, a longer time should be expected for the first cycle, as all the received files are new to xGitGuard and they all will be processed. In the second cycle and after, the previously processed files will not be processed (unless they have been modified) and it will take shorter time for xGitGuard to process the queries.

5.2. Git Hook

In this approach, xGitGuard is used as a hook integrated with GH. This approach will specifically be applicable on GitHub enterprise (GHE). The hook is responsible for intercepting new commits and processing them immediately for any hardcoded secrets. Depending on security operations center's (SOC) strategy, the commits containing secrets can be dropped or not.

5.3. BigQuery GH Datasets

GitHub has over 100 million repos with only ~30 million repos being public. The rest of repos on public GH are licensed. It is wise to scan not only the public GH repos, but also the licensed repos. GH provides weekly snapshots of open-sourced licensed repos that can be queried by Google BigQuery [11]. In this scenario, the scanned repos can be passed to xGitGuard for secret detection.

6. Conclusion

We developed an internal tool called xGitGuard to detect leaked secrets in open source code repositories, to keep our data safe from inadvertent leaks. It was designed to perform with a high level of agility and a low false positive rate. Its strength, compared to other methods using regular expression scanning, is its use of NLP and text processing to detect secrets, and machine learning/ML to validate the detections. Our in-house developed algorithms and technologies outperform existing solutions in this area. xGitGuard is a stand-alone model that can be used in different ways depending on the use case. If well-trained, the ML-based validator has a high level of accuracy. Such high accuracy reduces the rate of false positives.

Abbreviations

E	Extensions
GH	GitHub
GHE	GitHub Enterprise
ML	Machine Learning

NLP	Natural Language Processing
PII	Personally Identifiable Information
PKEY	Primary Keyword
SKEY	Secondary Keyword
SOC	Security Operations Center
xGG	xGitGuard

Bibliography & References

[1] GitHub, "GitGub About," [Online]. Available: https://github.com/about. [Accessed 8 July 2021].

[2] GitHub, "The 2020 State of the Octoverse," [Online]. Available: https://octoverse.github.com/. [Accessed 9 July 2021].

[3] Github Kills Search After Hundreds of Private Keys Exposed, [Online]. Available: https://it.slashdot.org/story/13/01/25/132203/github-kills-search-after-hundreds-of-private-keys-exposed. [Accessed 10 July 2021].

[4] M. Jackson, "Biggest Security Takeaway of 2020: Don't Leak Secrets on GitHub," DZone, [Online]. Available: https://dzone.com/articles/automating-secrets-detection-with-git-hooks. [Accessed 10 July 2021].

[5] T. Security, "Truffle Hot," 2021. [Online]. Available: https://github.com/trufflesecurity/truffleHog.

[6] T. Security, Truffle Security, [Online]. Available: https://github.com/trufflesecurity/truffleHog. [Accessed 2021].

[7] Nightfall, "Detect sensitive data in your GitHub repos," [Online]. Available: https://try.nightfall.ai/radar. [Accessed 8 July 2021].

[8] S. Lounici, "Optimizing Leak Detection in Open-source Platforms with Machine Learning Techniques," in *7th International Conference on Information Systems Security and Privacy*, Vienna, Austria, 2021.

[9] M. Mouw, "Profiling the abuse of exposed secrets in public repositories".

[10] American Express, "EarlyBird," 2021. [Online]. Available: https://github.com/americanexpress/earlybird. [Accessed 8 July 2021].

[11] Google BigQuery, "GitHub BigQuery Dataset," [Online]. Available: https://console.cloud.google.com/marketplace/details/github/github-repos. [Accessed 8 July 2021].

