International Journal of Engineering and Techniques - Volume 10 Issue 2, March 2024

RESEARCH ARTICLE OPEN ACCESS

Uncovering Sensitive Data Exposure: Analyzing JavaScript Files and GitHub Repositories

Mehank Prakash Kadu, Kusumlata Dhiman
Computer Science and Engineering, Parul University/PIT, Vadodara
Email: m3hank1337@gmail.com
Computer Science and Engineering, Parul University/PIT, Vadodara
Email: kusumdhiman212@gmail.com

Abstract:

DorkSearcher is a tool that streamlines bug hunting by efficiently searching for sensitive information like API keys and tokens in JavaScript files and GitHub repositories. It uses multithreading and advanced search techniques to automate vulnerability detection, enhancing security practices. With a user-friendly interface and customizable search options, DorkSearcher empowers teams to proactively address security risks and strengthen software application security.

Keywords — **Automation, Tokens, API Keys, Tokens, JavaScript Files, Multithreading, Bug-Hunting, Github, Sensitive Data.**

I. INTRODUCTION

DorkSearcher is a crucial tool in today's cybersecurity landscape, addressing the pressing issue of developers inadvertently exposing sensitive information on GitHub repositories. With the growing trend of developers integrating APIs and managing credentials within their codebase, the risk of unintentional data exposure has heightened significantly. DorkSearcher steps in as a proactive solution, offering a streamlined approach to recursively search directories for user-provided dorks—specific patterns or keywords indicative of sensitive data such as API keys, tokens, or hardcoded credentials. This tool harnesses the power of regular expressions (regex) to efficiently match dorks against source code files, enabling the rapid detection of potentially exposed information that could pose security risks if left unaddressed. By empowering users to conduct comprehensive scans and identify vulnerable areas within their software projects, DorkSearcher plays a pivotal role in bolstering data security and mitigating the risks associated with inadvertent information disclosure on GitHub repositories.

I. METHODOLOGY

A. Overview of DorkSearcher

DorkSearcher represents a sophisticated solution engineered with a singular focus on automating the detection and mitigation of vulnerabilities concerning hardcoded API keys, tokens, and sensitive data within software repositories and codebases. The tool's design philosophy revolves around enhancing security practices by empowering users to proactively identify and remediate potential security risks before they can be exploited. With its advanced capabilities and user-friendly interface, DorkSearcher stands as a pivotal asset in bolstering data security and fortifying software projects against potential threats.

B. Development Environment

DorkSearcher represents a sophisticated solution engineered with a singular focus on automating the detection and mitigation of vulnerabilities concerning hardcoded API keys, tokens, and sensitive data within software repositories and codebases. The tool's design philosophy revolves around enhancing security practices by empowering users to proactively identify and remediate potential security risks before they can be exploited. With its advanced capabilities and user-friendly interface, DorkSearcher stands as a pivotal asset in bolstering

data security and fortifying software projects against potential threats.

C. Algorithmic Approach

The fundamental algorithm powering DorkSearcher leverages sophisticated pattern-matching techniques, employing regular expressions (regex), to pinpoint hardcoded API keys, tokens, and patterns indicative of sensitive data within source code files. This algorithmic framework is designed to offer flexibility and precision, supporting both exact word matching and case-insensitive search options tailored to user preferences. By harnessing the capabilities of regex, DorkSearcher streamlines the identification of potential security vulnerabilities, enabling users to conduct thorough scans and proactively address sensitive data exposures within their software projects.

D. Multithreading Implementation

To improve scanning efficiency and performance, DorkSearcher utilizes the ThreadPoolExecutor from the concurrent.futures module for multithreaded processing. This enables concurrent scanning of multiple files, significantly reducing the overall scanning time for large codebases.

```
└─$ dorker -h
usage: dork-improved.py [-h] [--dorks dorks [dorks ...]] [--dork-file dork_file] [--max-threads max_
                        [-ft file_types [file_types ...]] [-mw] [-c context] [-cs]
                        directory_path

Search for specified 'dorks' in files within a directory and its subdirectories.

positional arguments:
  directory_path        Path to the directory to start the operation

options:
  -h, --help            show this help message and exit
  --dorks dorks [dorks ...]
                        List of search terms (dorks) to find in the files
  --dork-file dork_file
                        Path to a text file containing a list of dorks
  --max-threads max_threads
                        Maximum number of threads
  -ft file_types [file_types ...], --file-types file_types [file_types ...]
                        List of file extensions to include in the search (e.g., .py .php .txt)
  -mw, --exact-word     Match the exact word only
  -c context, --context context
                        Number of characters around the dork to print
  -cs, --case-sensitive
                        Enable case-sensitive search
```

Fig: 1 Customizable scanning options in DorkSearcher

E. Technologies and Usage Commands:

1. **Programming Language:** DorkSearcher is primarily developed using Python, a versatile and widely used programming language known for its readability and extensive libraries. Python's built-in support for regular expressions (regex) makes it an ideal choice for implementing pattern-matching techniques essential for identifying sensitive data patterns within source code files.

2. **Command-Line Interface (CLI):** DorkSearcher features a user-friendly Command-Line Interface (CLI) that simplifies interaction with the tool. Users can execute various commands to initiate scans, specify search criteria (dorks), configure scanning options (exact word matching, case sensitivity), generate reports, and manage scanning sessions efficiently.

3. **Usage Commands:** Below are some examples of usage commands in DorkSearcher:

    1. **Scan for API_KEY using command-line argument:**
       `python3 dorker.py [directory to search] --dorks "AfIl_KEY"`

    2. **Scan for dorks listed in a file using command-line argument:**
       `python3 dorker.py [directory to search] --dork-file dorks.txt`

    3. **Usage Command for Scanning Only Specific File Types:**
       `python3 dorker.py [directory to search] --dorks "AfIl_KEY" --file-types .py, .php, .txt`

III. RESULTS AND ANALYSIS

A. Functionality Testing

Single Dork Search:

1. DorkSearcher successfully identified occurrences of the specified single dork

keyword within source code files across multiple directories and subdirectories.

2. The tool provided accurate results, highlighting the line numbers and file paths where the dork was found.

Dork File Search:

1. When provided with a dork file containing multiple dork keywords, DorkSearcher efficiently scanned and located all keywords recursively in the specified path.
2. The tool displayed comprehensive results, indicating the occurrences of each keyword along with relevant context and file information.

B. Performance Evaluation:

The performance of DorkSearcher in recursive dork searches was evaluated based on the following metrics:

Scanning Speed:

1. DorkSearcher demonstrated rapid scanning speed, capable of searching through large codebases and directories within seconds.

Accuracy:

1. The tool maintained high accuracy in Identifying and reporting occurrences of dork keywords, ensuring minimal false positives.

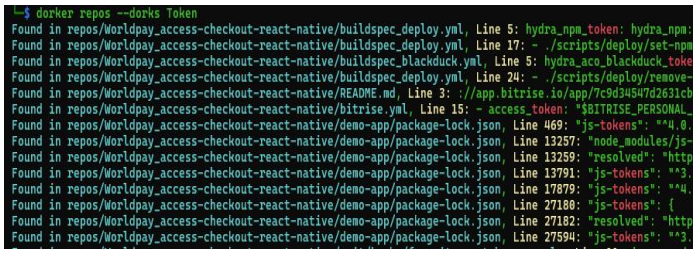

Fig: 2 Initiating a scan for Token keyword in a specified directory.

IV. CONCLUSION

In conclusion, DorkSearcher stands out as an indispensable tool for uncovering and addressing vulnerabilities such as hardcoded credentials, API keys, tokens, and sensitive data within software repositories. Its robust scanning capabilities, combined with customizable options and user-friendly interface, enable organizations to proactively identify and mitigate security risks. By swiftly detecting and highlighting potential exploits, DorkSearcher empowers security teams to fortify their defences and safeguard against unauthorized access, ultimately contributing to heightened cybersecurity resilience in today's dynamic digital environment.

V. REFERENCES

1. *MITRE. (n.d.). CWE-798: Use of Hard-coded Credentials. Retrieved from https://cwe.mitre.org/data/definitions/798.html*
2. *Kumar, C. (2023, May 15). How to Scan GitHub Repository for Credentials? Retrieved from https://geekflare.com/github-credentials-scanner/*
3. *Truffle Security. (n.d.). Introducing TruffleHog. Retrieved from https://trufflesecurity.com*
4. *Atyat, O. (2020, December 23). Your Full Map to GitHub Recon and Leaks Exposure. Medium. Retrieved from https://orwaatyat.medium.com/your-full-map-to-github-recon-and-leaks-exposure-860c37ca2c82*
5. *Shinde, A. (2023, June 24). Exploiting Exposed Tokens and API Keys - Edition 2023. Medium. Retrieved from https://kongsec.medium.com/exploiting-exposed-tokens-and-api-keys-edition-2023-e894a3af7dcd*
6. *Shinde, A. (2023). How to JS for Pentest: Edition 2023. Medium. Retrieved from https://kongsec.medium.com/how-to-js-for-bug-bounties-edition-2023-7108b56d9db6*
7. *Bugcrowd. (n.d.). GitHub Recon and Sensitive Data Exposure Module. Bugcrowd. Retrieved from https://www.bugcrowd.com/resources/levelup/github-recon-and-sensitive-data-exposure-module/*

8. *Gowtham. (n.d.). Sensitive information using GitHub. GitBook. Retrieved from https://gowthams.gitbook.io/bughunter-handbook/list-of-vulnerabilities-bugs/recon-and-osint/sensitive-information-using-github*

9. *Ghumade, S. (n.d.). Scanning JS Files for Endpoints and Secrets. Security Junky. Retrieved from https://securityjunky.com/scanning-js-files-for-endpoint-and-secrets/*

10. *Ponnusamy, P. (n.d.). Find the treasure hidden inside JavaScript. Medium. Retrieved from https://pravinponnusamy.medium.com/find-the-treasure-hidden-in-javascript-546827e1a4e2*