**MINOR CYBERSECURITY PROJECT**

**Detection of Leaked Secrets Using GitDorker & Gitleaks**

**Submitted by:** Premsai Dulla

**Institution / Course:** IIIT KOTTAYAM/CYBER SECURITY **Date:** 06/08/2025

**🧭 Introduction**

This mini-project focuses on understanding how leaked secrets (e.g., API keys, tokens) can be discovered from public GitHub repositories. To simulate a real-world scenario in a safe and ethical manner, a dummy GitHub organization was created and seeded with fake repositories containing test secrets.

**🎯 Objective**

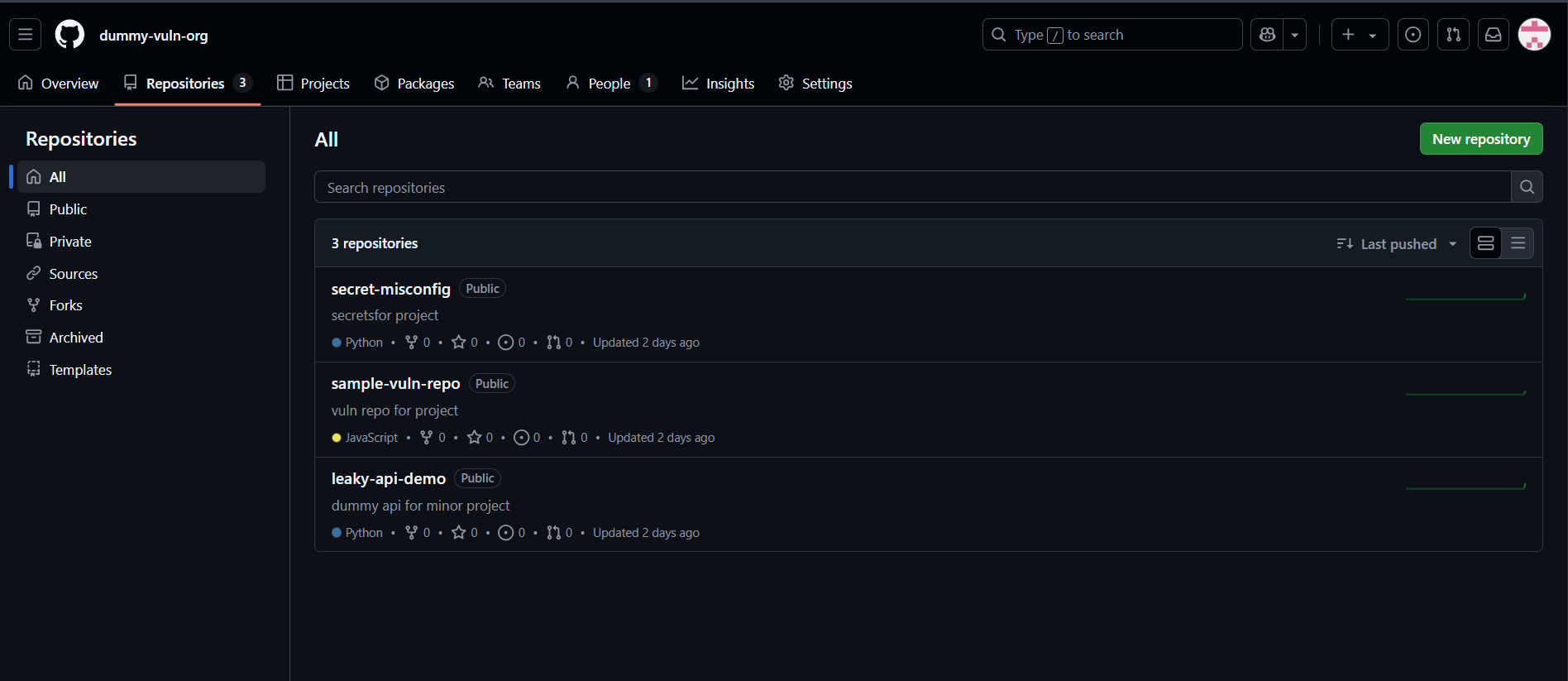
To detect leaked credentials and API keys within a targeted GitHub organization using GitDorker and Gitleaks, demonstrating how such exposure can be identified and mitigated.

**🛠️ Setup**

**Dummy Organization**

* Name: dummy-vuln-org
* Link: https://github.com/dummy-vuln-org
* Contains 3 public repositories:
  + secret-misconfig
  + sample-vuln-repo
  + leaky-api-demo

Each repository includes fake secrets for detection and analysis purposes.



**🔍 Tools Used**

| **Tool** | **Purpose** |
| --- | --- |
| GitDorker | Scans for exposed credentials via Google dorks and GitHub search |
| Gitleaks | Detects secrets and credentials via pattern-matching in source code |
|  |  |

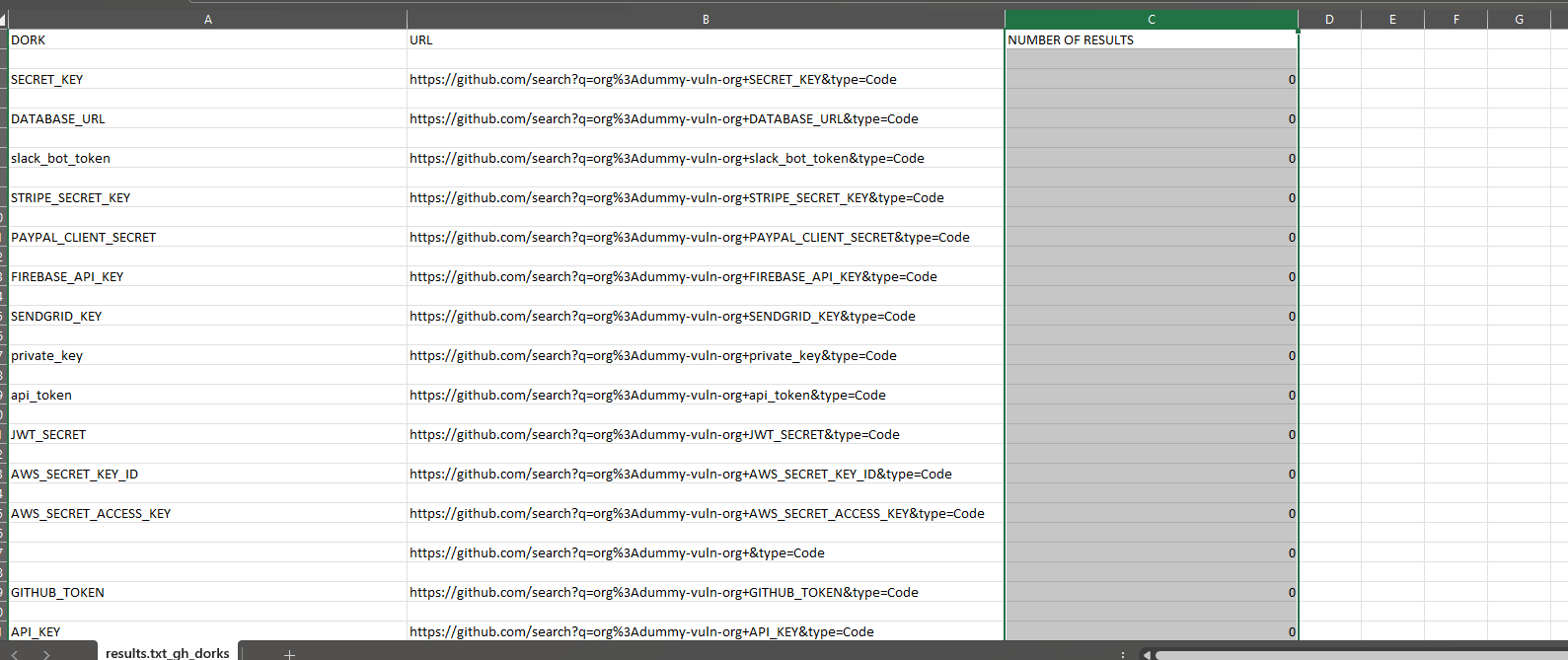
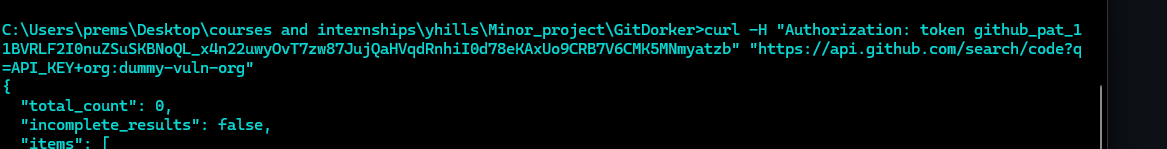
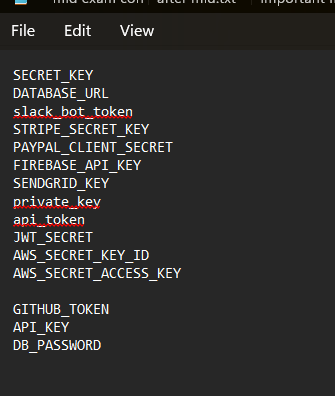
**🚀 Methodology**

**Step 1: GitDorker Scanning**

Executed the following command:

bash

**python GitDorker.py -d dorks.txt -q "org:dummy-vuln-org" -tf tokens.txt -o results.txt**

* dorks.txt contained relevant search keywords
* tokens.txt held valid GitHub tokens
* Output: results.txt generated but no matches found (likely due to GitHub indexing delays)
* 
* Check whether it is github problem (indexing) or local machine :
* 
* **DORKS.txt**
* 

**Step 2: Gitleaks Detection**

Used local scanning for each cloned repository:

**gitleaks detect --source=. --report-path=gitleaks-report.json --report-format=json**

Each repository yielded JSON reports with discovered secrets based on detection rules like:

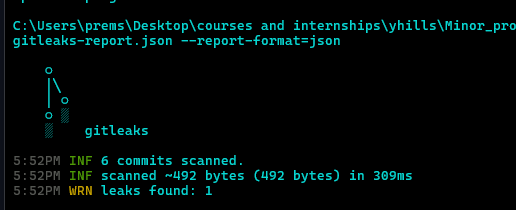
* Generic API keys
* AWS credentials
* Private tokens

**📂 Results & Findings**

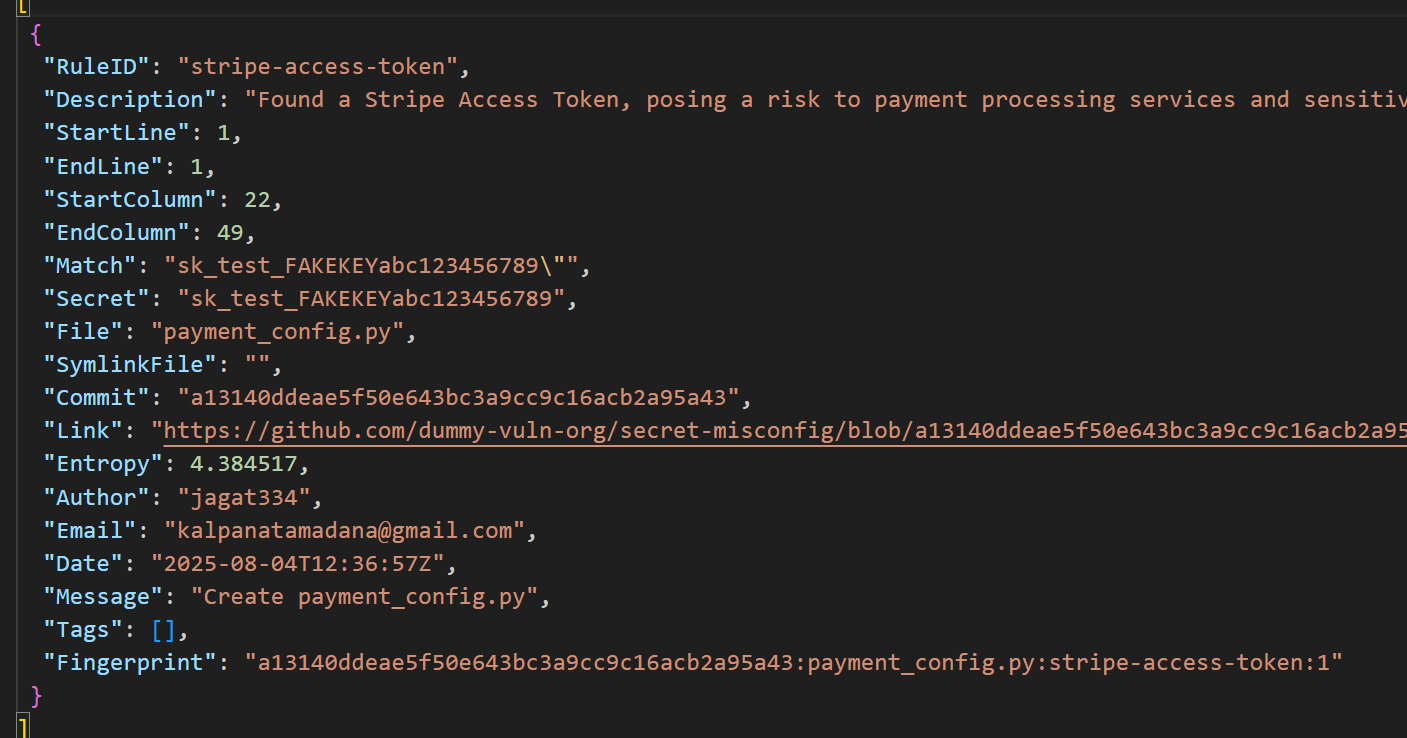
Each repository’s scan revealed fake secrets such as:

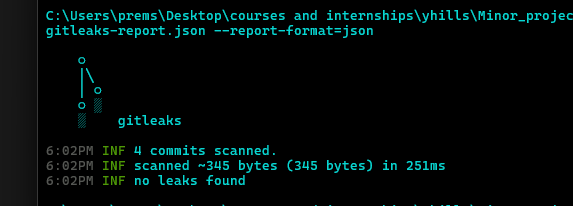
* Hardcoded tokens in config files
* AWS-like keys embedded in scripts
* Misconfigured .env files with fake credentials

**Secret-misconfig repo :**

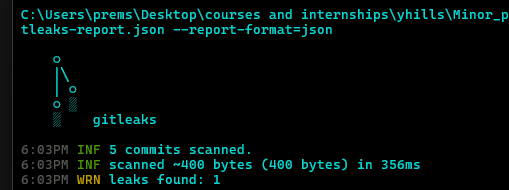


**Jsonformat :**

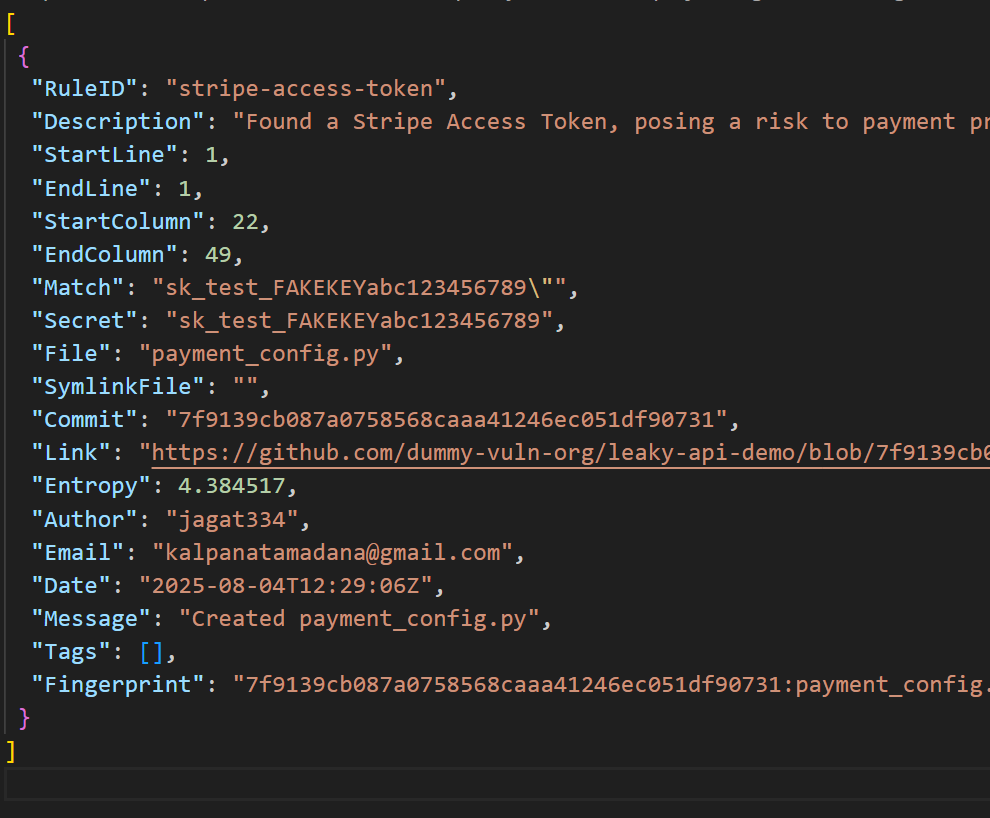


Sample-vuln-repo : 

**Leaky-api-demo:**



**Jsonformat :**



Reports stored in gitleaks/ folder and excluded from Git commits using .gitignore.

**🚫 Precautions & Best Practices**

**To ensure sensitive data wasn’t pushed to GitHub:**

* **Added this to .gitignore:**
* **gitleaks/**
* **gitleaks/secret-misconfig**
* **gitleaks/sample-vuln-repo**
* **gitleaks/leaky-api-demo**
* Removed from tracking (if previously committed):

bash

**git rm --cached -r gitleaks/**

**✅ Conclusion**

This project showcases how easily secrets can leak through source code — even in public repositories. It highlights the value of tools like Gitleaks in securing codebases and encourages proactive scanning during development to prevent data exposure.