GitHub OSINT – Mini Cybersecurity Project Report

Name: Uday Kiran

Date: 30-07-2025

# 1. Introduction

In the modern software development world, publicly hosted repositories such as GitHub are often unintentionally used as a medium for exposing sensitive data. This project focused on Open Source Intelligence (OSINT) gathering from public GitHub repositories, with the aim to discover leaked credentials, API tokens, and secrets using a combination of GitHub dorks and automated tools.

# 2. Tools Used

- Custom GitHub Dorks – Manual search queries to find exposed secrets in code and files.

- GitLeaks (v8.28.0) – An automated tool for scanning Git repositories for secrets like API keys, tokens, and passwords.

# 3. Methodology

## Step 1: Target Selection

Selected various public GitHub repositories such as:  
- HariSekhon/Kubernetes-configs  
- maxh33/showcase-twitter-clone  
- fingerprintjs/fingerprintjs-pro-server-api-node-sdk

## Step 2: Manual Dorking

Used the GitHub search bar to run queries such as:  
- filename:password.ini  
- kong\_admin password  
- vercel token

## Step 3: GitLeaks Scan

Cloned the selected repository locally and ran the following command:

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

# 4. Key Findings

## Finding 1: Hardcoded Kong Credentials in Public Repository

Repo: HariSekhon/Kubernetes-configs  
Credential: kong\_admin / password  
Risk: High – Default credentials could be used to access sensitive admin panels if left unchanged.

## Finding 2: Vercel and Deployment Secret Metadata Exposed

Repo: maxh33/showcase-twitter-clone  
Secrets Mentioned: VERCEL\_TOKEN, JWT\_SECRET\_KEY, EMAIL\_HOST\_PASSWORD, etc.  
Risk: High – If any values are committed, they can be exploited by attackers.

## Finding 3: Placeholder Secrets in Public Code

Repo: fingerprintjs/fingerprintjs-pro-server-api-node-sdk  
Secret Example: <SECRET\_API\_KEY>  
Risk: Medium – Even placeholders reveal potential attack surfaces.

## Finding 4: Git Leaks Scan Result

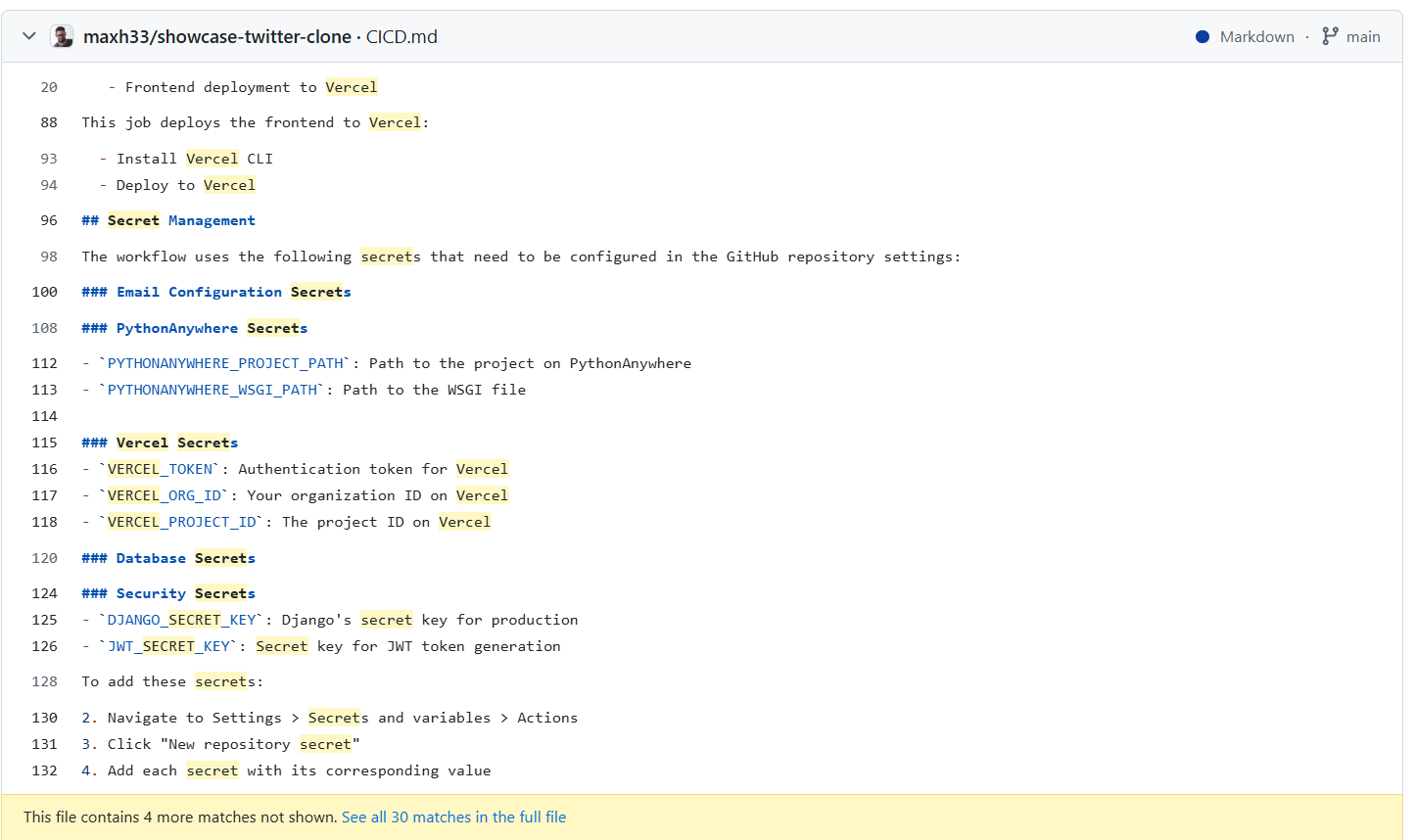
Tool Used: Git Leaks v8.28.0  
Repo Scanned: showcase-twitter-clone  
Result: Secrets detected and logged in gitleaks-report.json  
Risk Level: Medium/High depending on the secrets found

***5. Screenshots***

Screenshot 1: Hardcoded Kong credentials in repo README

A close-up of a computer screen

AI-generated content may be incorrect.

Screenshot 2: GitHub Actions YAML exposing secret variable names

Screenshot 3: Placeholder API keys in sample code

A screenshot of a computer

AI-generated content may be incorrect.

# 6. Recommendations

- Never commit secrets, passwords, or tokens to public repositories.  
- Always add `.env`, `.ini`, or config files to `.gitignore`.  
- Use tools like GitLeaks and GitHub Secret Scanning to detect leaks early.  
- Rotate exposed secrets immediately.  
- Apply the OWASP password storage best practices.

# 7. Conclusion

This mini project successfully identified several real-world examples of poor secret management in public GitHub repositories. The use of GitHub dorking and GitLeaks proved effective in uncovering potentially exploitable issues. This highlights the need for strong DevSecOps hygiene and automated secret scanning tools in the software development lifecycle.