

GitHub Recon

— and what you can achieve with
it!



The Open Security Community



Who am I?

- Undergraduate **Computer Engineering** Student
- Full-stack Web Developer (**LAMP/LEMP/JAM**)
- **Web/Network** Penetration Tester
- HIGHLY Passionate **CTF Player**
- **Community Administrator** at Ask Buddie
- **Advocate** at **Hacking is NOT a Crime**

Recon Highlights:

- Implementing **Active** and **Passive Reconnaissance** in **Penetration Testing** and **Bug Hunting**,
- Familiarity and enthusiasm towards **Intelligence Gathering** disciplines like **OSINT** and **GEOINT**!

Not specifically focused into Information Security!

RECONNAISSANCE

All about inspecting or exploring about the target!

RECON in Information Security

- ❑ Exploring about the target to gather confidential information,
- ❑ Endless treasure of information that can lead to successful attacks,
- ❑ Figuring out the internal workflow of the target without actually being associated with the target,
- ❑ Finding out the information required to get unauthorized access to certain asset of the target,
- ❑ Carrying out attacks in a stealthy and precise manner!

What can you achieve?

- Subdomains
- Virtual Hosts
- SSH credentials
- Database details
- Source codes
- Open ports and services
- Hidden endpoints
- WHOIS, IP and DNS Info
- User account details
- Third-party associations

What is GitHub Recon?

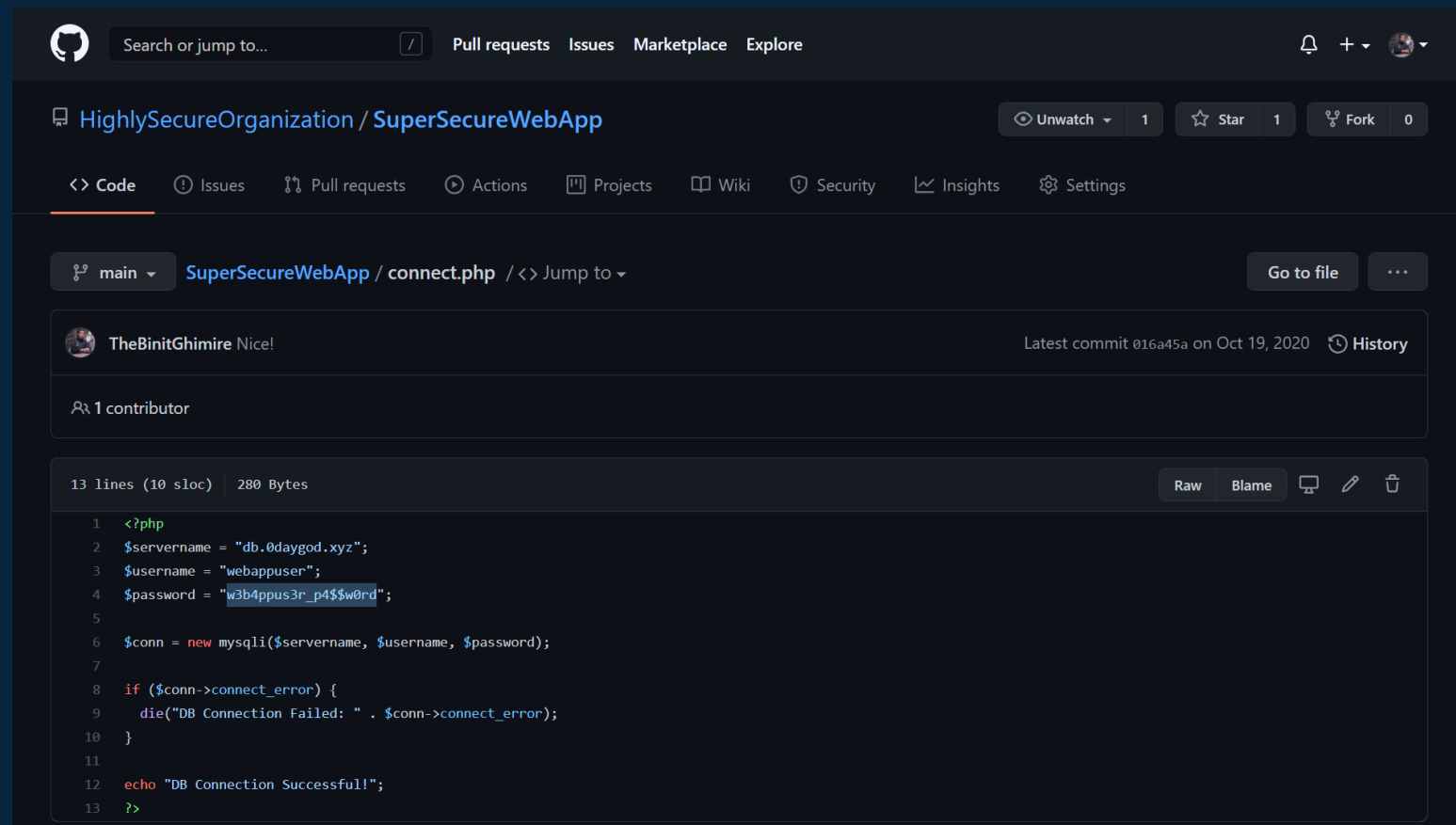


- ❑ Doing RECON with the help of GitHub,
- ❑ Finding out essential information using several features offered within GitHub

GitHub for Reconnaissance

- ❑ Finding out sensitive information disclosures from a target repository,
- ❑ Using specialized keywords to gather information associated with the target,
- ❑ Iterating through the commit history in the target's repositories to figure out any way to obtain unintended information or access to certain asset!

Sensitive Information Disclosures on GitHub



The screenshot displays the GitHub interface for the repository 'HighlySecureOrganization / SuperSecureWebApp'. The file 'connect.php' is open, showing 13 lines of PHP code. The code contains hardcoded database credentials, including a password that is partially highlighted in blue. The interface includes navigation tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. The file's commit history and contributor information are also visible.

```
1 <?php
2 $servername = "db.0daygod.xyz";
3 $username = "webappuser";
4 $password = "w3b4ppus3r_p4$$w0rd";
5
6 $conn = new mysqli($servername, $username, $password);
7
8 if ($conn->connect_error) {
9     die("DB Connection Failed: " . $conn->connect_error);
10 }
11
12 echo "DB Connection Successfull!";
13 ?>
```


GitHub Dorks for Recon

org:<organization>

language:<language>

user:<username>

filename:<file-name>

extension:<extension>

<specific-keyword>

"<multiple-words>"

path:<file-path>

<dork>:"<multiple-words>"

<dork> <keyword>

Sample Keywords to search for

- password
- API_KEY
- APP_ID
- AUTH_TOKEN
- AUTH_KEY
- AWS_SECRET
- AWS_SECRET_KEY
- AWS_ACCESS_KEY
- AUTH
- AUTH0_CLIENT_ID
- AUTH0_CLIENT_SECRET
- CARGO_TOKEN
- CF_PASSWORD
- CI_USER_TOKEN
- DATABASE_PASSWORD
- DOCKER_HUB_PASSWORD
- ELASTICSEARCH_PASSWORD
- email
- EXP_PASSWORD
- FIREBASE_API_TOKEN
- FTP_LOGIN
- FTP_PASSWORD
- GH_AUTH_TOKEN
- id_rsa.pub
- JWT_SECRET
- jdbc_user
- mailchimp_api_key
- MANIFEST_APP_TOKEN
- MYSQL_PASSWORD
- NETLIFY_API_KEY
- NPM_API_TOKEN
- OSSRH_JIRA_PASSWORD
- passwordTravis
- s3_access_key
- SENDGRID_KEY
- SSMTP_CONFIG
- TRAVIS_SECURE_ENV_VARS
- TWILIO_TOKEN
- URBAN_MASTER_SECRET
- VIP_GITHUB_DEPLOY_KEY
- WORDPRESS_DB_PASSWORD
- YT_CLIENT_SECRET

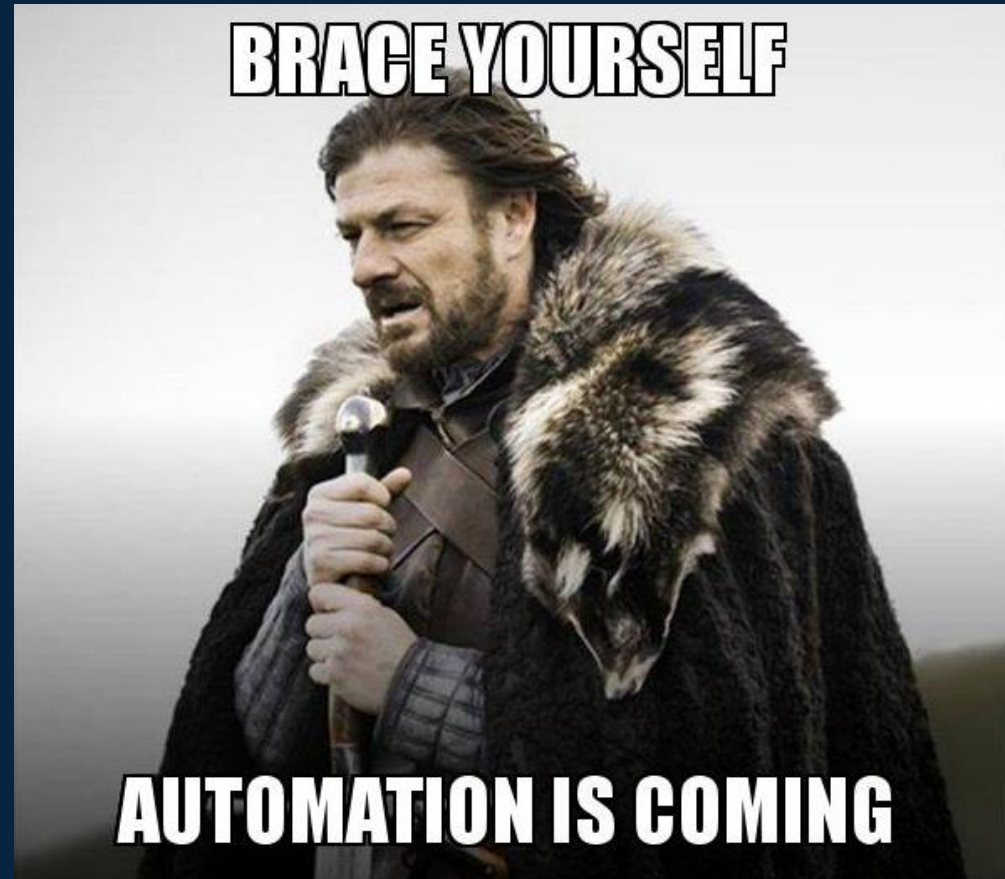


DEMONSTRATION

Using **GitHub Dorks** for Recon!

Automated GitHub Recon

- ❑ ~~GitRob~~ (wraith)
- ❑ truffleHog
- ❑ git-secrets



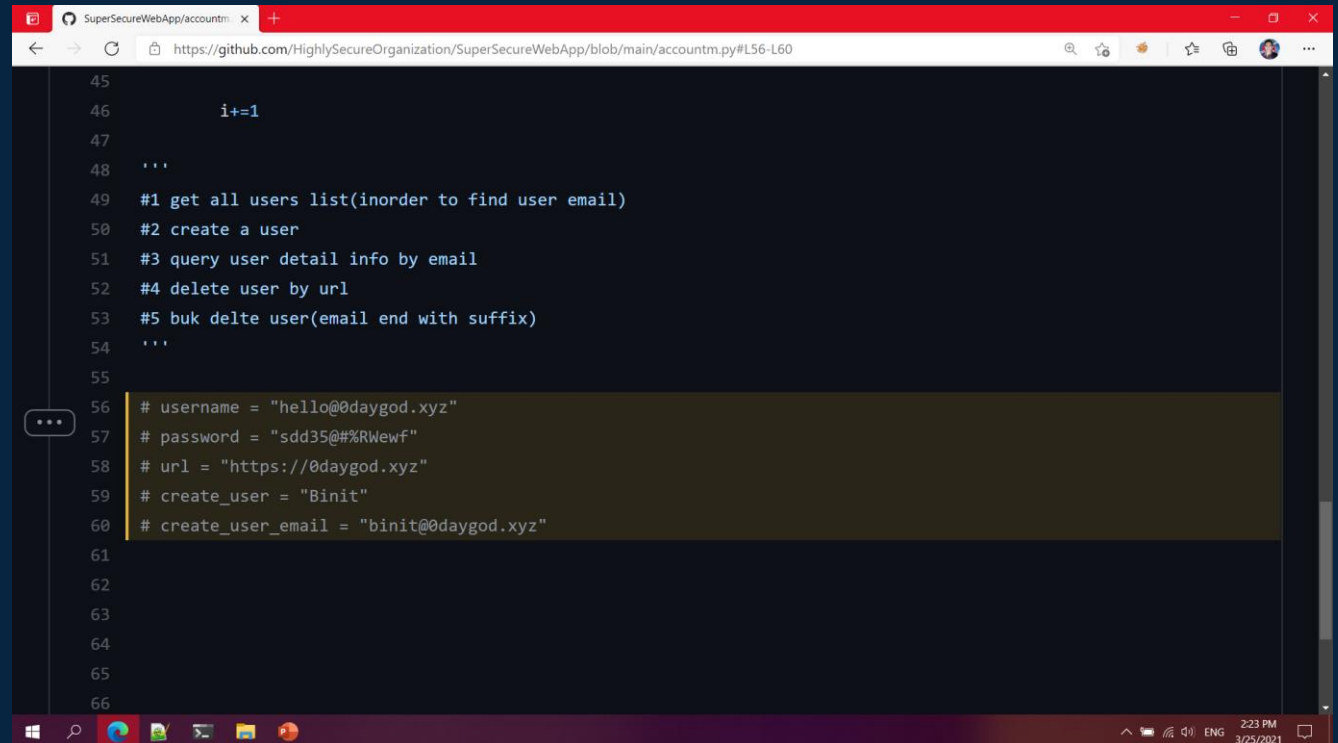


DEMONSTRATION

truffleHog and **git-secrets**!

Preventing GitHub Leaks

- ❑ Using .gitignore
- ❑ GitHub Secrets



The screenshot shows a web browser window displaying a GitHub repository. The URL in the address bar is <https://github.com/HighlySecureOrganization/SuperSecureWebApp/blob/main/accountm.py#L56-L60>. The code is a Python file named `accountm.py`. Lines 45-55 show a loop and a list of API endpoints. Lines 56-60 show hardcoded credentials for a user creation process, which are highlighted with a yellow background:

```
45  
46     i+=1  
47  
48     ...  
49     #1 get all users list(inorder to find user email)  
50     #2 create a user  
51     #3 query user detail info by email  
52     #4 delete user by url  
53     #5 buk delte user(email end with suffix)  
54     ...  
55  
56     # username = "hello@0daygod.xyz"  
57     # password = "sdd35@%RWef"  
58     # url = "https://0daygod.xyz"  
59     # create_user = "Binit"  
60     # create_user_email = "binit@0daygod.xyz"  
61  
62  
63  
64  
65  
66
```

The Windows taskbar at the bottom shows the time as 2:23 PM on 3/25/2021.

Using .gitignore

```
root@localhost: ~/git/SuperSec  Windows PowerShell
PS F:\git\HighlySecureOrganization\SuperSecureWebApp> cat .\helloworld.php
<?php

$username = "hello";
$password = "world";

?>
PS F:\git\HighlySecureOrganization\SuperSecureWebApp> cat .\.gitignore
helloworld.php
PS F:\git\HighlySecureOrganization\SuperSecureWebApp> git add .\helloworld.php
The following paths are ignored by one of your .gitignore files:
helloworld.php
Use -f if you really want to add them.
PS F:\git\HighlySecureOrganization\SuperSecureWebApp>
```

GitHub Secrets

SECURITY

Are you storing your secret keys, passwords and other sensitive information in your codes? What if you want to push your codes to GitHub? Would you be exposing them to the public? Majority of the open-source data leaks happen as a result of sensitive information disclosure on platforms like GitHub, GitLab, BitBucket, etc.

So, if you are interested in preventing such data leaks and exposures while pushing your codes to GitHub, start making use of **GitHub Secrets** right now! You can easily find the option to store such information using **GitHub Secrets** from your repository settings, and secrets are just encrypted environment variables which are only made available to specific actions.

Are **U** secure?



GitHub Secrets




Secrets

New secret

Secrets are environment variables that are **encrypted** and only exposed to selected actions. Anyone with **collaborator** access to this repository can use these secrets in a workflow.

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more](#).

Repository secrets

 DBHOST	Updated 13 hours ago	<button>Update</button>	<button>Remove</button>
 DBPASSWORD	Updated 13 hours ago	<button>Update</button>	<button>Remove</button>
 DBUSER	Updated 13 hours ago	<button>Update</button>	<button>Remove</button>

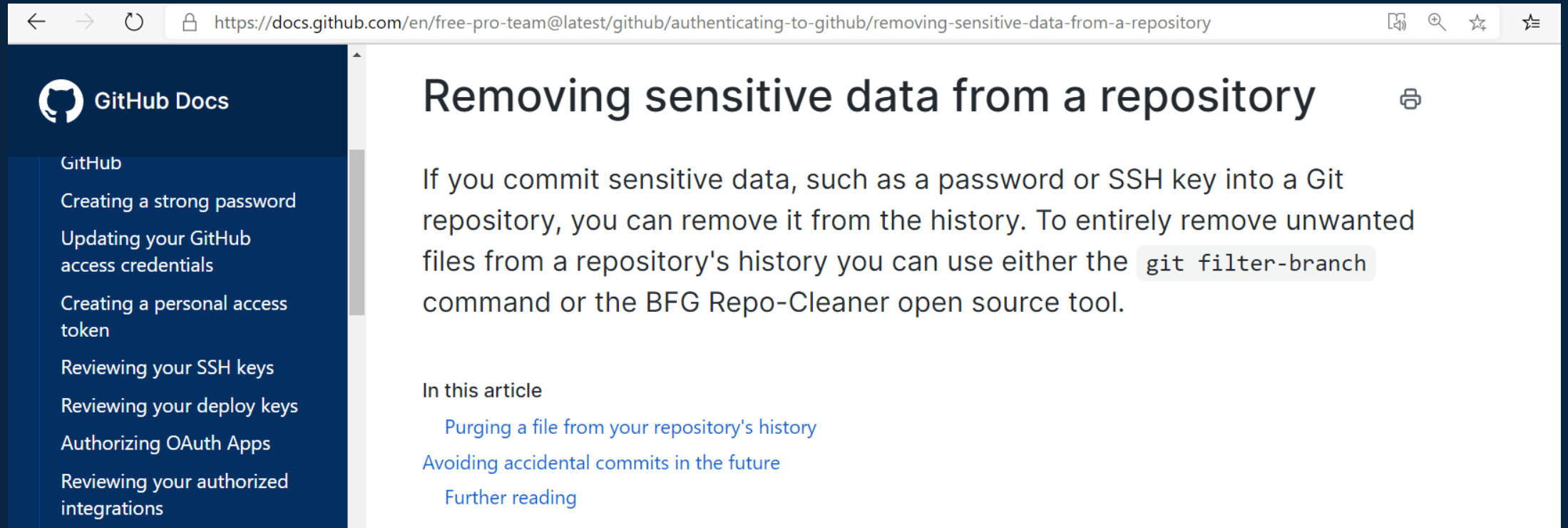
← → ↺ 🔒 https://github.com/HighlySecureOrganization/SuperSecureWebApp/blob/main/includes/conn.php

13 lines (10 sloc) | 303 Bytes

```
1 <?php
2 $servername = "${{ secrets.DBhost }}";
3 $username = "${{ secrets.DBuser }}";
4 $password = "${{ secrets.DBpassword }}";
5
6 $conn = new mysqli($servername, $username, $password);
7
8 if ($conn->connect_error) {
9     die("DB Connection Failed: " . $conn->connect_error);
10 }
11
```




Removing Leaks



The screenshot shows a web browser displaying the GitHub documentation page for 'Removing sensitive data from a repository'. The browser's address bar shows the URL: <https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/removing-sensitive-data-from-a-repository>. The page has a dark blue sidebar on the left with the GitHub logo and a list of navigation links. The main content area is white and features the article title, a brief introduction, and a list of links for further reading.

← → ↺ 🔒 <https://docs.github.com/en/free-pro-team@latest/github/authenticating-to-github/removing-sensitive-data-from-a-repository> 📖 🔍 ☆ ⌵

 GitHub Docs

- GitHub
- Creating a strong password
- Updating your GitHub access credentials
- Creating a personal access token
- Reviewing your SSH keys
- Reviewing your deploy keys
- Authorizing OAuth Apps
- Reviewing your authorized integrations

Removing sensitive data from a repository

If you commit sensitive data, such as a password or SSH key into a Git repository, you can remove it from the history. To entirely remove unwanted files from a repository's history you can use either the `git filter-branch` command or the BFG Repo-Cleaner open source tool.

In this article

- [Purging a file from your repository's history](#)
- [Avoiding accidental commits in the future](#)
- [Further reading](#)



<https://WHOISbinit.me/>

Do you have any queries?

THANK YOU!

Let me know if you need the slides or resources!



InternetHeroBINIT



thebinitghimire



WHOISbinit



thebinitghimire