

# GitHub Recon

– and what you can achieve with it!



Presented by  DigitalOcean + intel + 



# 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

## 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?

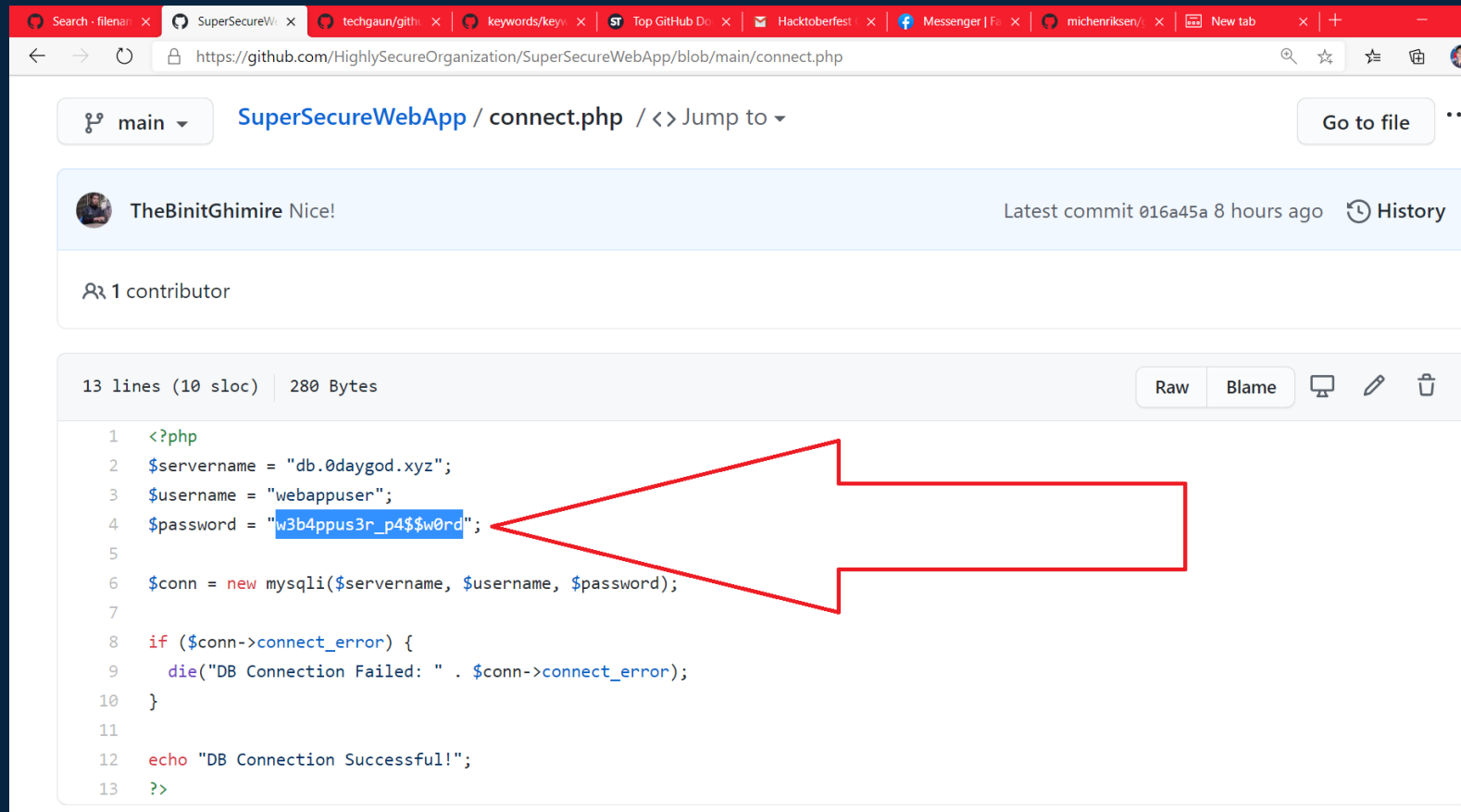


- ❑ Doing RECON with the help of GitHub,
- ❑ Finding out essential information using several features offered within GitHub

# 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 shows a web browser displaying a GitHub repository page for 'SuperSecureWebApp'. The file 'connect.php' is open, showing 13 lines of PHP code. A red arrow points to line 4, where a password is hardcoded: `$password = "w3b4ppus3r_p4$$w0rd";`. The password is highlighted in blue. The browser's address bar shows the URL `https://github.com/HighlySecureOrganization/SuperSecureWebApp/blob/main/connect.php`. The repository page includes a commit message 'Nice!' by 'TheBinitGhimire' and a 'Go to file' button.

```
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 Successful!";
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

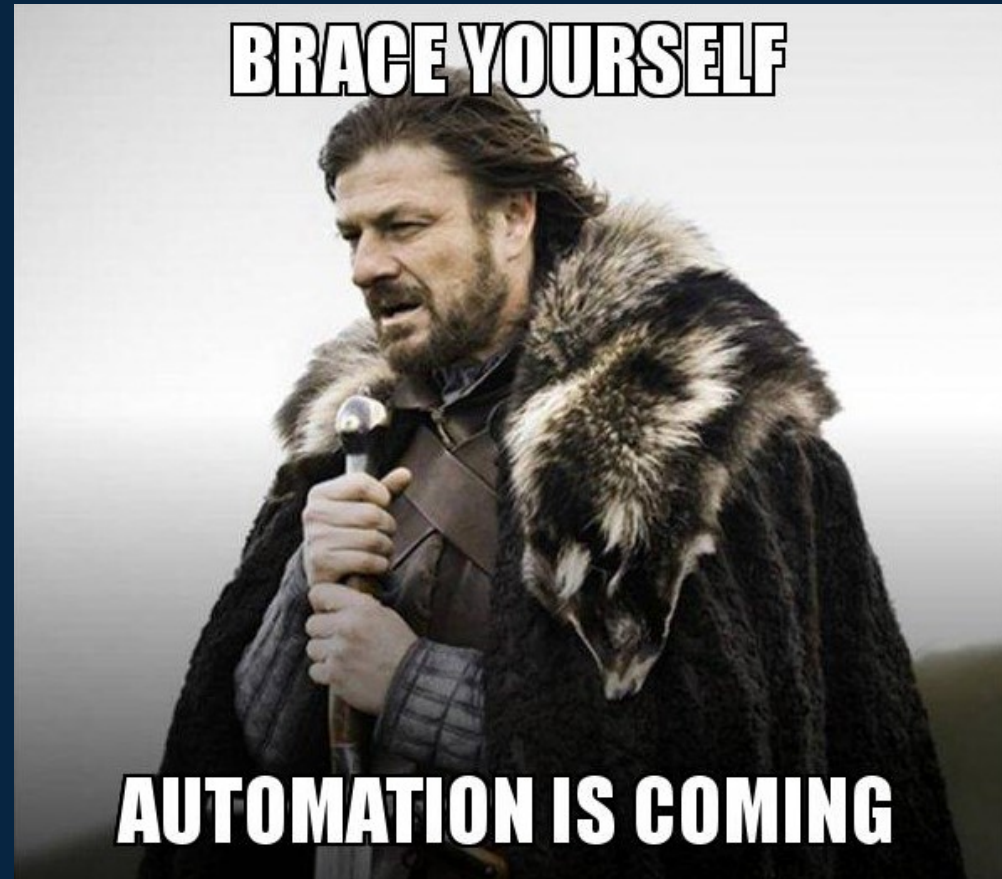


# DEMONSTRATION

Using **GitHub Dorks** for Recon!

# Automated GitHub Recon

- ❑ GitRob
- ❑ truffleHog
- ❑ git-secrets





# DEMONSTRATION

**GitRob, truffleHog and git-secrets!**



# Preventing GitHub Leaks

- ❑ Using .gitignore
- ❑ GitHub Secrets



The screenshot shows a web browser displaying a GitHub repository page. The URL in the address bar is `https://github.com/HighlySecureOrganization/SuperSecureWebApp/blob/main/accountm.py`. The page shows a Python file named `accountm.py` with line numbers 54 through 64. A black rectangular box highlights the code between lines 56 and 60, which contains hardcoded credentials:

```
56 # username = "hello@0daygod.xyz"
57 # password = "sdd35@#%RWewf"
58 # url = "https://0daygod.xyz"
59 # create_user = "Binit"
60 # create_user_email = "binit@0daygod.xyz"
```

# 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>

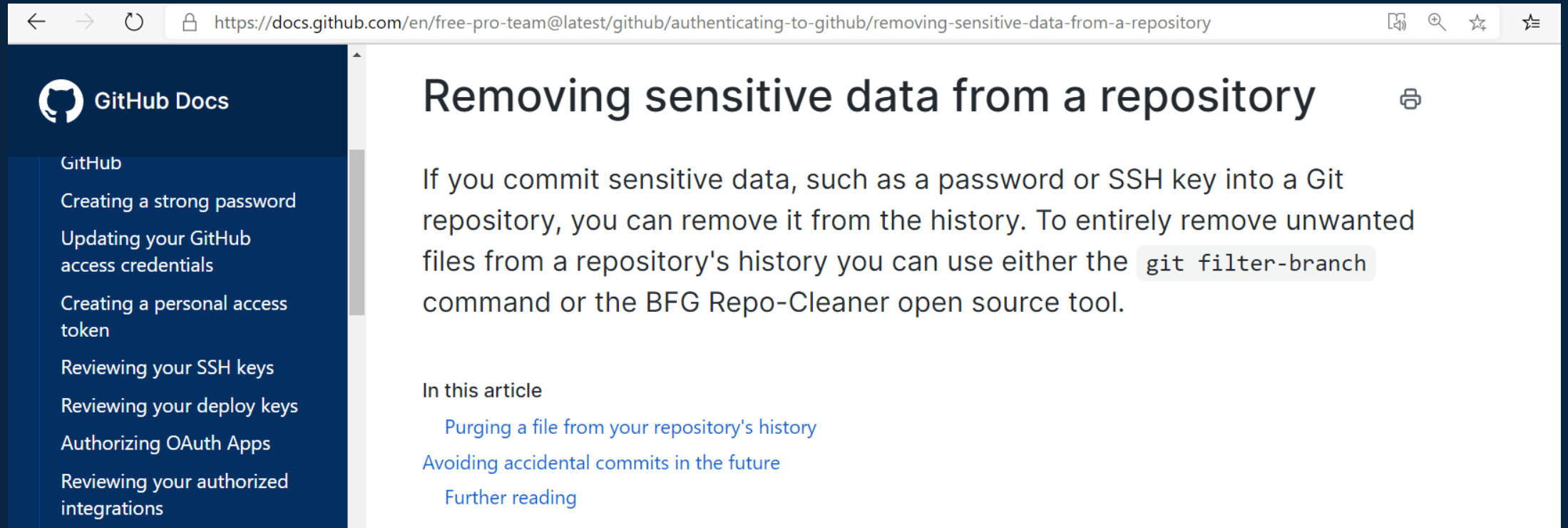
https://github.com/HighlySecureOrganization/SuperSecureWebApp/blob/main/includes/conn.php

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 features a dark blue sidebar with the GitHub logo and a list of navigation links. The main content area has a white background with the article title and a brief introduction. The introduction states that sensitive data like passwords or SSH keys can be removed from a repository's history using either the `git filter-branch` command or the BFG Repo-Cleaner tool. Below the introduction, there is a section titled 'In this article' with three links: 'Purging a file from your repository's history', 'Avoiding accidental commits in the future', and '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