Hunting for secrets in bug bounty targets

BY BLACKBIRD-EU · OCTOBER 5, 2025 · LAST UPDATED ON OCTOBER 12, 2025

It is no secret that bug bounty hunters who spend lots of time on information gathering are always rewarded well for their efforts. As developers continue to in-deliberately push secrets to production or to other public-facing resources, hunting for secrets remains invaluable for security researchers like us. Especially as a single pair of credentials could introduce a whole new attack surface for us to explore.

In this article, we dive into how you can discover secrets through various ways to score more bounties on your favorite programs.

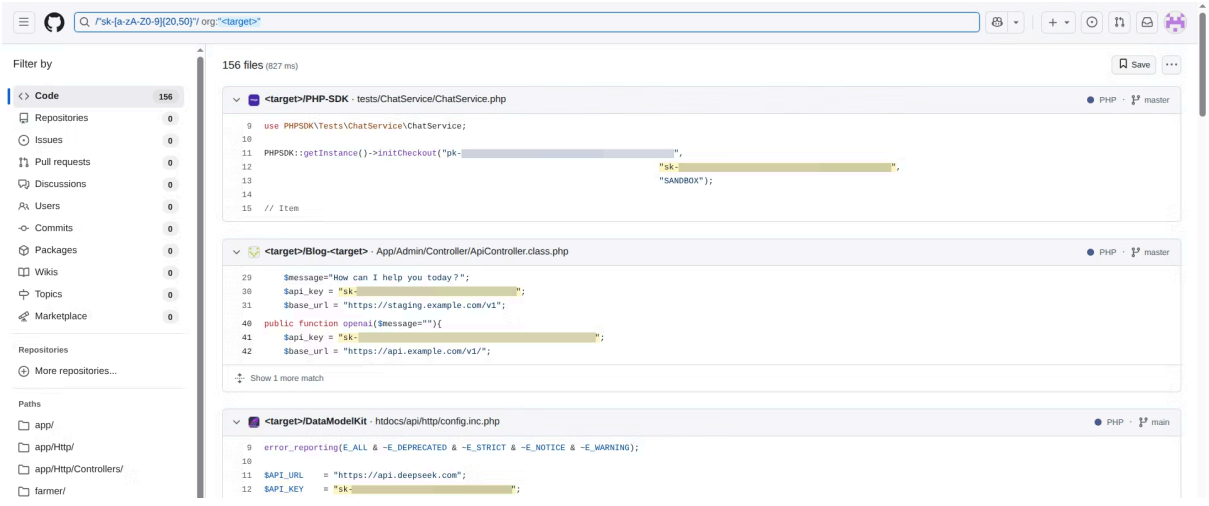
Let's dive in!

GitHub dorking

Code collaboration platforms like GitHub and GitLab enable software development teams to effortlessly work together on projects. Unfortunately, it still happens that developers accidentally push new code commits with hard-coded secrets and only notice when they've already been made public.

On the bright side, this opens up an opportunity for us to look for these hard-coded secrets. This act of information gathering by browsing code commits is also referred to as GitHub dorking. This method of searching involves narrowing down our keywords to a single organization or user while filtering for common patterns.

Let's take a look at an example:



Hunting for OpenAI API keys using GitHub search (dorking)

In the example above, we narrowed down our search to our target organization. Additionally, we also included a keyword to filter for OpenAI API keys. OpenAI is a service that's widely used by organizations to integrate AI into their existing tech stack. Finding such tokens can help us get access to their OpenAI account and possibly consume credits or read training data.

Below is a short list with more examples, each with its respective description:

```
org:"example" /"sk-[a-zA-Z0-9]{20,50}"/                                  # Hard-coded OpenAI API key
org:"example" (AWS_ACCESS_KEY_ID OR AWS_ACCESS_SECRET_KEY)               # Hard-coded AWS access & secret key
org:"example" ("sk_live_" OR "pk_live_")                                 # Hard-coded Stripe secret keys
org:"example" (SENDGRID_API_KEY OR sendgrid_api_key)                     # SendGrid API keys
org:"example" (ANTHROPIC_API_KEY OR anthropic_api_key)                   # Anthropic API keys
org:"example" (PAYPAL_CLIENT_SECRET OR paypal_client_secret)             # PayPal credentials
org:"example" (SQUARE_ACCESS_TOKEN OR square_access_token)               # Square payment tokens
org:"example" (AZURE_CLIENT_SECRET OR AZURE_CLIENT_ID)                   # Azure credentials
org:"example" (CLOUDFLARE_API_TOKEN OR CF_API_TOKEN)                     # Cloudflare tokens
org:"example" (filename:.env OR filename:.env.local OR filename:travis.yml)  # Configuration and build files
org:"example" /http(s)?:\/\//                                            # Hard-coded links
org:"example" ("mongodb://" OR "mongodb+srv://" OR "mysql://")           # Database connection strings
org:"example" ("jwt_secret" OR "JWT_SECRET" OR "jwtSecret")              # Authentication & security tokens
org:"example" (extension:pem OR extension:key OR extension:p12 OR extension:pfx)  # Certificate files
org:"example" (SLACK_BOT_TOKEN OR SLACK_WEBHOOK_URL)                     # Slack integration tokens
org:"example" (GITHUB_TOKEN OR GITHUB_PAT OR GH_TOKEN)                   # GitHub personal access tokens
org:"example" /\/\/(.*\.)?amazonaws\.com/                                # AWS endpoints
org:"example" /\/\/(.*\.)?firebaseio\.com/                               # Firebase endpoints
```

Finding more vulnerabilities with GitHub dorking

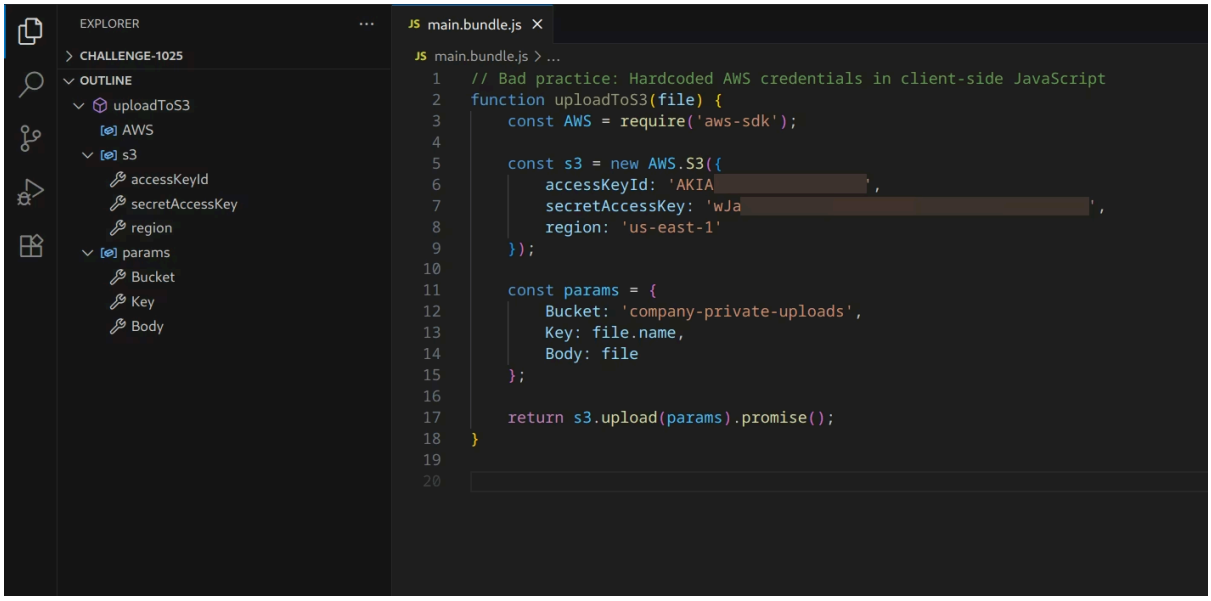
Want to score more bounties through GitHub dorking? Make sure you give our comprehensive GitHub dorking guide a read!

https://www.intigriti.com/researchers/blog/hacking-tools/advanced-github-dorking-guide

JavaScript files

JavaScript files are goldmines for bug bounty hunters looking for security vulnerabilities. Modern web applications heavily rely on JavaScript nowadays, which often inadvertently expose references to undocumented application routes, API endpoints, and input parameters, authentication tokens (such as AWS credentials), application logic, and configuration details that should remain hidden.

These secrets can often help us expand our attack surface or even lead to direct vulnerabilities (such as DOM-based vulnerabilities). For instance, it happens that a developer unknowingly hard-codes AWS credentials in a function call:



Hunting for AWS keys using JavaScript file enumeration

If we can manage to map out all JavaScript files of a target, we could, in practice, run an automated tool to grep for common patterns that would catch secrets like in the example above.

Our next step is validating these secrets. API keys and authentication tokens can often easily be confirmed by navigating the product's documentation. Open-source repositories like Streaak's Keyhacks can also aid in validating secrets. The key here is to ensure the found credential is indeed a secret and can help elevate standard privileges, as some tokens and API keys are meant to be public.

Score more bounties with JavaScript files

JavaScript files are goldmines for bug bounty hunters! In our comprehensive guide, we show you how to spot vulnerabilities by performing JavaScript analysis. Read the article now:

https://www.intigriti.com/researchers/blog/hacking-tools/testing-javascript-files-for-bug-bounty-hunters

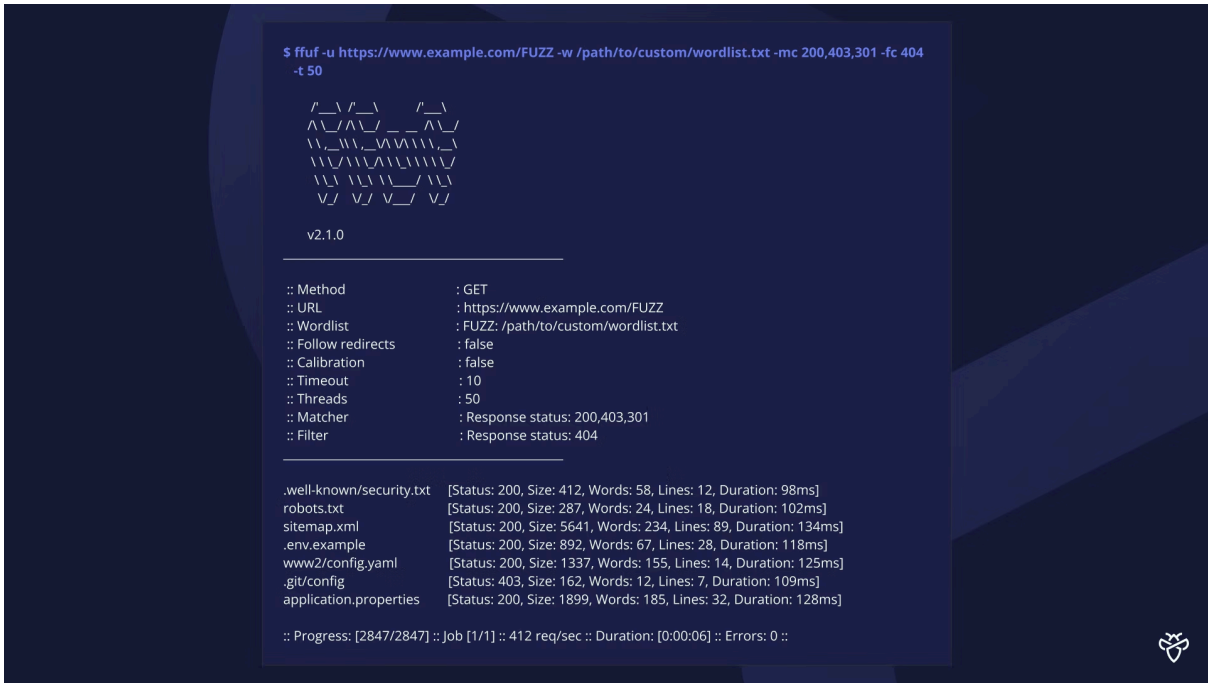
Exposed configuration files

Configuration files help centralize sensitive information like database credentials, API keys, encryption secrets (such as SSH key pairs), service endpoints, etc. The issue is that these files can appear anywhere. For instance, through misconfigured web servers, lack of access controls on certain directories and paths, and even in forgotten backup copies.

For security researchers like us, the hardest part is knowing where to look for these sensitive config files. Let's take a look at a few methods we can apply to find files like these.

Content bruteforcing

Bruteforcing filenames on the target server is one of the most reliable methods for discovering hidden configuration files. The key is in using comprehensive, custom wordlists that match our target's utilized technologies. Open-source tooling like Ffuf and Dirbuster can easily help us discover assets that we failed to discover through other methods.



Content discovery (bruteforcing) with custom wordlists

> **Bruteforcing with custom wordlists**
>
> In our detailed guide, we've outlined how you can create your custom wordlist for your target to find more assets.
>
> https://www.intigriti.com/researchers/blog/hacking-tools/creating-custom-wordlists-for-bug-bounty-targets-a-complete-guide

Google dorking

Search engines like Google and Bing are constantly crawling and indexing websites, and sometimes index files that developers never intended to be public. This makes it interesting for us as we can leverage built-in search operators to find exposed configuration files that have been indexed.

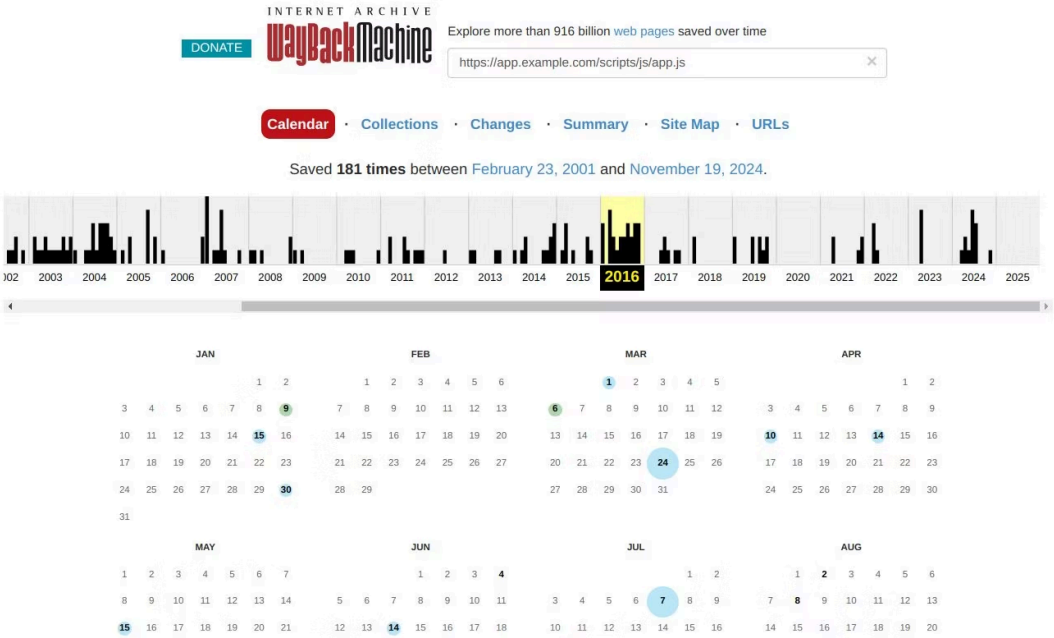
> **Finding more vulnerabilities with Google**
>
> If you wish to learn more about how you can leverage Google and Bing search to find more vulnerabilities, make sure you read our comprehensive article on this topic:
>
> https://www.intigriti.com/researchers/blog/hacking-tools/google-dorking-for-beginners-how-to-find-more-vulnerabilities-using-google-search

Internet archive

Similar to search engines, Internet archives index websites constantly, too, but also save versions in the form of 'snapshots' of individual pages on the Internet. These snapshots can be valuable as resources that are no longer available now, and can still be viewed via an Internet archiving service.

For instance, if hard-coded credentials have been briefly exposed in a configuration or JavaScript file, and subsequently removed a few days later. It may happen that these are still viewable by retrieving the correct snapshot.

This makes it worth checking out internet archiving services like the Wayback Machine or CommonCrawl for archived versions of your target:



Example of a legacy version of a JavaScript file

Conclusion

Performing reconnaissance always pays off, as it can help you discover possible secrets and expand your initial attack surface. The tricky part is understanding what and where to look for them. In our article, we've detailed several methods to identify and validate disclosed secrets.

So, you've just learned something new about finding secrets in bug bounty programs... Right now, it's time to put your skills to the test! You can start by practising on vulnerable labs and CTFs or... browse through our 70+ public bug bounty programs on Intigriti, and who knows, maybe earn a bounty on your next submission!

START HACKING ON INTIGRITI TODAY