Testing JavaScript files for bug bounty hunters

BY BLACKBIRD-EU · DECEMBER 19, 2024 · LAST UPDATED ON MARCH 6, 2025

You've with no doubt heard or seen other fellow bug bounty hunters find critical vulnerabilities thanks to JavaScript file enumeration, right? This article is all about the importance of testing and examining JavaScript files for bug bounty hunters. We will guide you on what exactly to look for and provide examples of the most common vulnerabilities (and patterns that lead to vulnerabilities) in JavaScript files!

As with most of our technical content, you can always use this as your checklist whenever you're hunting for bugs on your bug bounty target!

Let's dive in!

Importance of JavaScript files in bug bounty

We all know how important JavaScript files are in bug bounty as they can result in a wide variety of (critical) vulnerabilities.

Some bug bounty hunters often overlook or even leave out testing and examining javascript files. Most commonly because they are hard to read and quite difficult to examine and fully understand.

However, these files often contain references to API endpoints, app routes and input parameters. Tokens or other hard-coded secrets can sometimes be also present in them.

For this reason, JavaScript files should never be neglected. Especially not when the target either looks outdated or is a JavaScript-heavy application.

In this article, we will guide you on what to look for in JavaScript files.

Interesting findings hidden in JavaScript files

Referenced API endpoints and app routes

JavaScript files contain general-purpose code (such as function calls) to make the application work for the end user. These functions, especially ones that are in place to communicate with an API (Application Programming Interface) have to define the endpoint to communicate with, including any input parameters or data structures that the API may expect.

Our main purpose as bug bounty hunters is to enumerate as many of them as possible as it's an integral part of our content discovery process.

Another thought to consider is, that some targets have API endpoints or app routes referenced in javascript files that are not part of the main application or are not intended for the end user (think of administrative panels or developer debugging tools). Browsing through the target with a proxy

intercepting tool may not pick up these hidden API endpoints or URLs, that's why we'd need to resort to additional ways of performing content discovery.

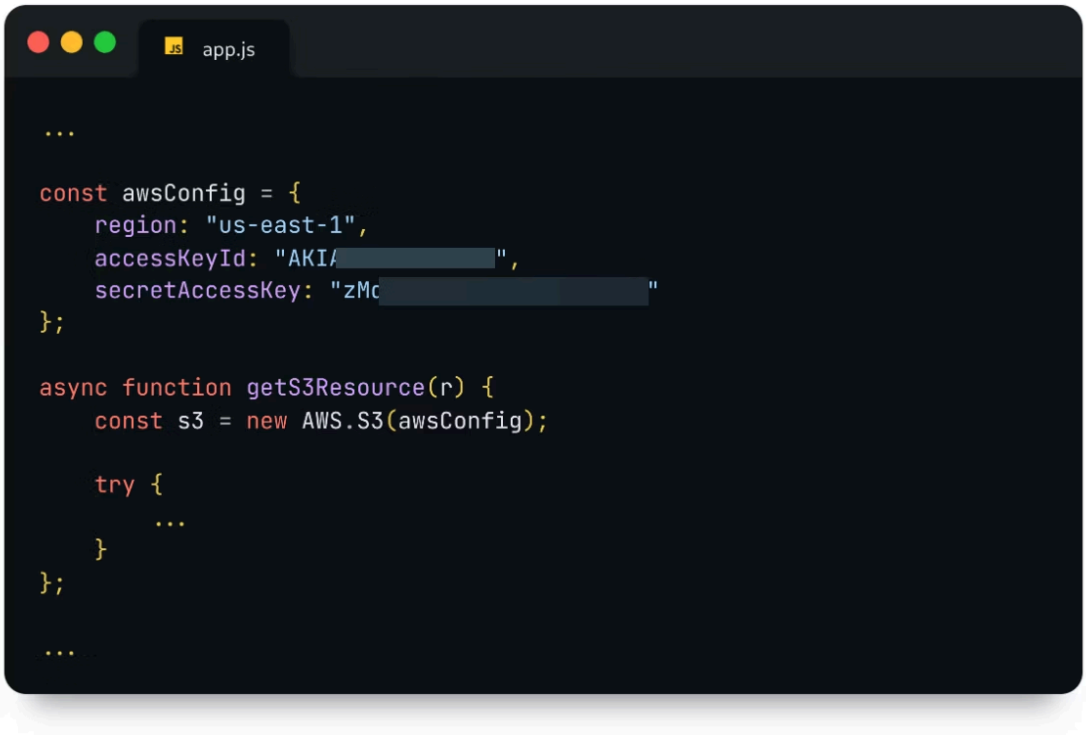
For this reason, you should always try finding these types of hard-to-reach API endpoints or application routes in JS files as they can severely increase our chances of discovering untested components that could lead to critical vulnerabilities.

TIP! LinkFinder by @GerbenJavado is an excellent and simple open-source tool to help you find links in JavaScript files!

Hard-coded credentials

It happens that developers accidentally make mistakes and commit hard-coded keys to public resources such as JavaScript files. For example, AWS credentials should not be hard-coded and present in static javascript files.

For this reason, it's recommended to look for any type of credential that is hard-coded in configuration files (often declared as global objects or variables).



Example of a configuration file with hard-coded AWS credentials

Automated tooling can pick up a vast amount of credential types, however, you will need to manually validate your discovered credentials before reporting as some of the results are either invalid or already revoked keys.

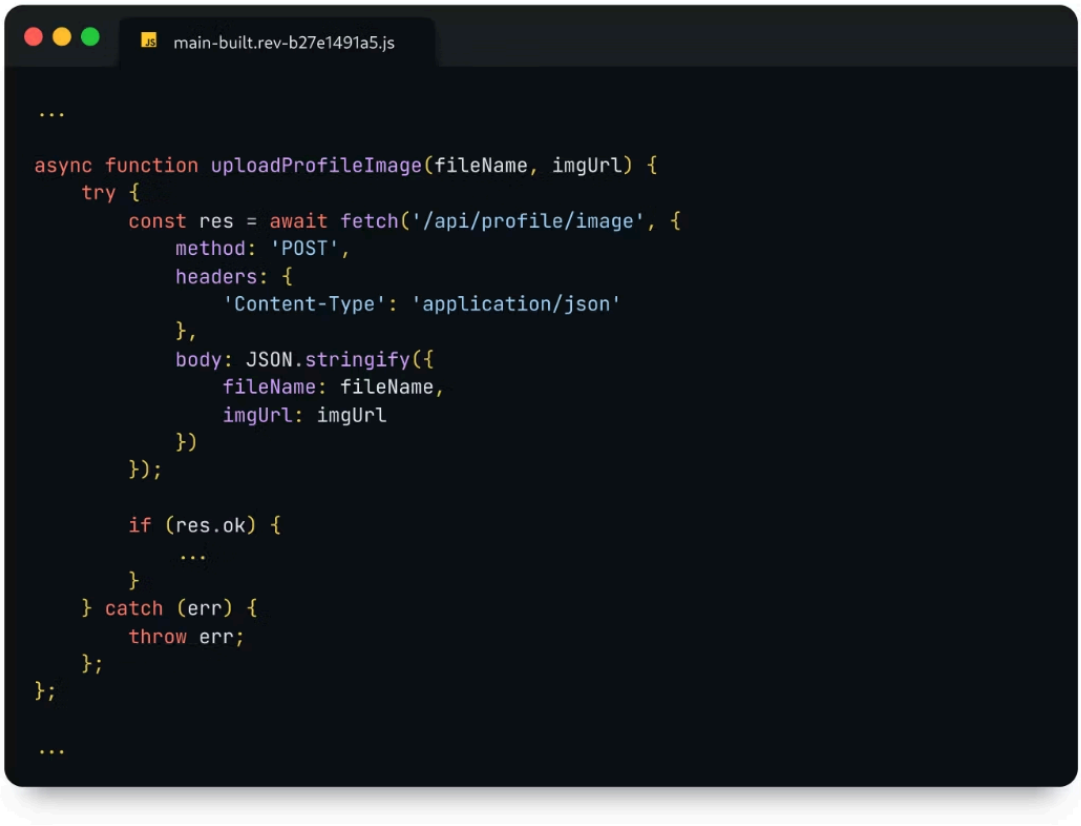
@Streaak has published an open-source GitHub repository that contains a list of popular public API keys and credentials and how to validate your discovered findings against these APIs.

TIP! Some API keys or tokens are meant to be public (e.g. Google API key). Always verify that the credentials are indeed not meant to be exposed and provide access to privileged actions.

Input parameters

Input parameters accept arbitrary user data and that can lead to all sorts of vulnerabilities, such as SSRF, SQL injections, XSS and much more. It's crucial that we map out all possible parameters that a particular endpoint may accept.

One quite accurate way to do so is via JavaScript files. In API function calls, you'll often find referenced body or query parameters:



Example code snippet with a function that references body parameters

Take note of these and try them in various ways against the API endpoint, try changing the content type of your request body as well. Remember that some parameters even get re-used.

Here's an additional tip by @Hakluke to help you find more XSS vulnerabilities!

DOM-based vulnerabilities

JavaScript files can also help find you more DOM-based vulnerabilities, such as DOM-based XSS or prototype pollution. DOM-based vulnerabilities require you to look for DOM sinks and sources. A DOM source is the source of where your arbitrary input comes from (such as `location.hash` or a query parameter). When the application is vulnerable, it passes your unsafe input directly to a DOM sink, a function call that supports dynamic client-side code execution (for example, functions like `eval`, `location.href` or `innerHTML`).

We can specifically look for these keywords to find DOM-based vulnerabilities. Prototype pollution vulnerabilities take the same approach and will require us to look for functions or methods that are merging objects with arbitrary inputs in an unsafe way.

Luckily for us, web extension tools like DOM Invader and Untrusted Types can help us automate the detection part right in our web browser. That way, we won't have to spend much time examining JS files manually!

Conclusion

JavaScript files should never be neglected, they contain valuable information that could help you find your next critical vulnerability! Always try to examine all embedded JS files within your target, including the ones that are dynamically imported after page load. JavaScript files can also help us expand our attack surface by specifically looking for referenced URLs, API endpoints and parameters.

You've just learned something new about JavaScript files and their importance in bug bounty hunting... Right now, it's time to put your skills to the test! Browse through our 70+ public bug bounty programs on Intigriti, and who knows, maybe your next bounty will be earned with us!

START HACKING ON INTIGRITI TODAY

REQUEST A DEMO

intigriti.com/demo

VISIT THE WEBSITE

intigriti.com

GET IN TOUCH

hello@intigriti.com