SARIF results file is too large

In this article

You cannot upload a SARIF results file larger than 10 MB to code scanning. Explore ways to generate a smaller file containing the highest impact results.

About this error

```
SARIF file is too large
SARIF results file is too large
SARIF upload is rejected (bigger than allowed size for zip archive)
SARIF ZIP upload is too large
A fatal error occurred: SARIF file is too large
413: Payload Too Large
```

One of these errors is reported if a process attempts to upload a SARIF file that is larger than the maximum size of 10 MB. Code scanning does not accept files above this size. There are several different ways to reduce the number of results generated for upload to code scanning.

You could see this error for SARIF files generated by CodeQL or by third-party analysis tools. For information about the limits on uploads, see code scanning, see "SARIF support for code scanning."

Confirming the cause of the error

There are many potential causes of very large SARIF results files.

SARIF file compression

Take a look at the results file that was rejected by code scanning to see if:

- the SARIF file was compressed using gzip
- the compressed file is smaller than 10 MB

If the file wasn't compressed using gzip, try compressing the file before rerunning the upload process. If the compressed file is still too large, you need to configure the analysis to generate a smaller set of results.

Amount of code analyzed

If you have too many results, you should configure analysis to analyze only the most

important code.

- For interpreted languages, check if the repository contains many tests, demos, or vendored dependencies where fixing alerts is a lower priority. Try excluding this code from analysis. For more information, see "Excluding code from analysis for interpreted languages."
- For compiled languages, check if the build process generates more than one variant of the code (for example, targets for multiple operating environments or architectures). Try analyzing just one variant of the code initially. For more information, see "Optimizing the build command."

Number of queries run

If you still have too many results, check how many queries you are using to analyze the code. Try running fewer queries. You can reintroduce additional queries when the initial alerts are fixed. For example, for CodeQL analysis you could run just the default suite of queries. For more information, see "Customizing your advanced setup for code scanning."

Number of results found by queries

Sometimes a single query reports many results because the codebase has a specific coding style, or because the analysis does not understand a particular library. You can review the results file in a SARIF viewer to see the distribution of results. For example, https://microsoft.github.io/sarif-web-component/.

- Check if the results are dominated by alerts identfied by a single query. Try excluding that query from analysis. You can reintroduce it when other alerts are fixed. For more information about CodeQL query configuration, see "Excluding a query from analysis."
- Check if there are dataflow queries with many deep paths. Try omitting dataflow paths from the output. For more information about CodeQL analysis configuration, see "Omitting dataflow paths from the output."

Fixing the problem

The following options are listed in order of complexity. You need to revise the configuration to reduce the number of results to a manageable size. Once you have fixed all of those alerts, you can update the configuration to expand the analysis to cover more code or run more queries.

Excluding code from analysis for interpreted languages

Excluding non-production code from analysis is a simple way to reduce the size of the results file.

- CodeQL advanced setup for code scanning: use `paths` and `paths-ignore` in the workflow file to specify what code to analyze. For more information, see "Customizing your advanced setup for code scanning."
- CodeQL CLI `database create`: create a YAML configuration file for code scanning using the same syntax to define which code to analyze. Update the `database create` command to call this configuration file using the `--codescanning-config` option. For more information, see "Customizing your advanced setup for code scanning."

Optimizing the build command

Using a build command that compiles only one variant is a simple way to reduce the size

of the results file.

- CodeQL advanced setup for code scanning: update the workflow file to specify your chosen build command. For more information, see "CodeQL code scanning for compiled languages."
- CodeQL CLI `database create`: specify your chosen build command either by calling the `database create` command with the `--command` option, or by defining the build command in a YAML configuration file for code scanning and calling the file using the `--codescanning-config` option. For more information, see "Preparing your code for CodeQL analysis."

Defining the query suite to run

You may already be running only the default security queries, but it is worth checking.

- CodeQL advanced setup for code scanning: check the workflow file for the `queries` keyword. If it is not present, then only the default query suite is run. If it is present, it defines which queries to run. Try commenting out this line of the workflow file. For more information, see "Customizing your advanced setup for code scanning."
- CodeQL CLI `database analyze`: check the database analysis command for any paths that specify queries, query suites, or query packs. If none are present, then only the default query suite is run. If any are present, they define which queries to run, you can try removing them from the call. For more information, see "Analyzing your code with CodeQL queries."

Excluding a query from analysis

If the results are dominated by the results for a single rule, excluding the rule from the analysis may be the best solution.

- CodeQL advanced setup for code scanning: use the `query-filters` keyword to exclude one or more queries from analysis. For more information, see "Customizing your advanced setup for code scanning."
- CodeQL CLI `database analyze`: update the database analysis command to exclude one or more queries from analysis. For more information, see "Analyzing your code with CodeQL queries."

Alternatively, you can use a tool like the filter-sarif action to rewrite the SARIF file to exclude specific detections via an exclusion pattern.

Omitting dataflow paths from the output

If there are many deep code paths highlighted in the SARIF results, you can reduce the number of paths reported for each alert.

- CodeQL advanced setup for code scanning: update the `analyze` step to limit the number of paths to a maximum of one or zero.

```
- name: Perform CodeQL Analysis
  uses: github/codeql-action/analyze@v2
  env:
    CODEQL_ACTION_EXTRA_OPTIONS: '{"database":{"interpret-results":["--max-
paths", 1]}}'
```

- CodeQL CLI `database analyze`: update the database analysis command to include the `--max-paths=1` flag. For more information, see "database analyze."

Note: The `max-paths` setting affects the results of all dataflow queries.

Further reading

- "SARIF support for code scanning"