Triaging code scanning alerts in pull requests

In this article

When code scanning identifies a problem in a pull request, you can review the highlighted code and resolve the alert.

> **Who can use this feature**
> If you have read permission for a repository, you can see annotations on pull requests. With write permission, you can see detailed information and resolve code scanning alerts for that repository.

Code scanning is available for organization-owned repositories in GitHub Enterprise Server. This feature requires a license for GitHub Advanced Security. For more information, see "About GitHub Advanced Security."

About code scanning results on pull requests

In repositories where code scanning is configured as a pull request check, code scanning checks the code in the pull request. By default, this is limited to pull requests that target the default branch, but you can change this configuration within GitHub Actions or in a third-party CI/CD system. If the lines of code changed in the pull request generate code scanning alerts, the alerts are reported in the following places on the pull request.

- Check results in the pull request
- The **Conversation** tab of the pull request, as part of a pull request review
- The **Files changed** tab of the pull request

If you have write permission for the repository, you can see any existing code scanning alerts on the **Security** tab. For information about repository alerts, see "Managing code scanning alerts for your repository."

In repositories where code scanning is configured to scan each time code is pushed, code scanning will also map the results to any open pull requests and add the alerts as annotations in the same places as other pull request checks. For more information, see "Customizing your advanced setup for code scanning."

If your pull request targets a protected branch that uses code scanning, and the repository owner has configured required status checks, then the "Code scanning results" check must pass before you can merge the pull request. For more information, see "About protected branches."

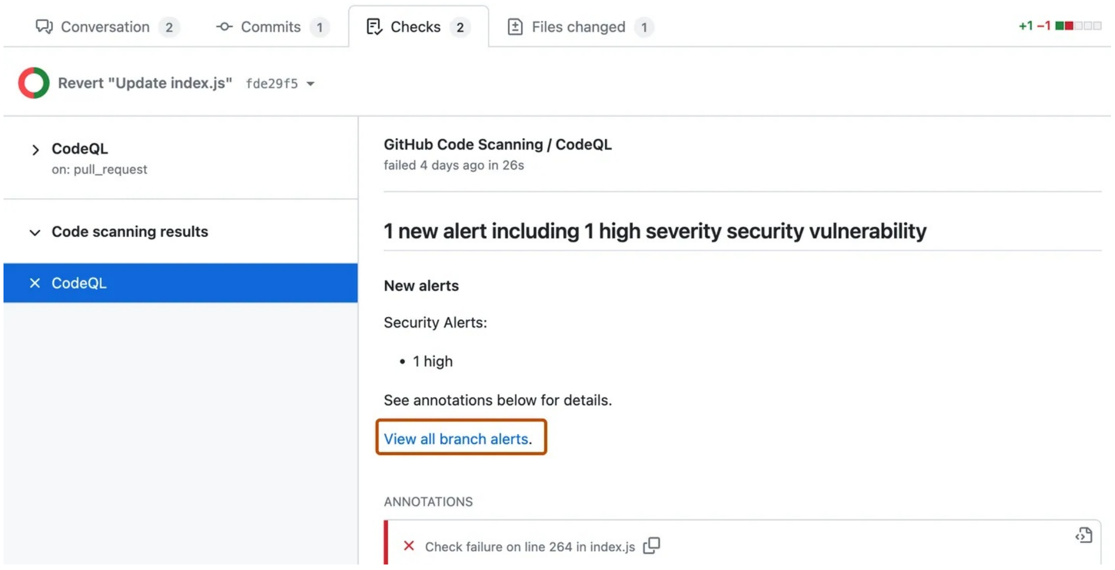
About code scanning as a pull request check

There are many options for configuring code scanning as a pull request check, so the exact configuration of each repository will vary and some will have more than one check.

Code scanning results check

For all configurations of code scanning, the check that contains the results of code scanning is: **Code scanning results**. The results for each analysis tool used are shown separately. Any new alerts on lines of code changed in the pull request are shown as annotations.

To see the full set of alerts for the analyzed branch, click **View all branch alerts**. This opens the full alert view where you can filter all the alerts on the branch by type, severity, tag, etc. For more information, see "Managing code scanning alerts for your repository."



GitHub Code Scanning / CodeQL
failed 4 days ago in 26s

1 new alert including 1 high severity security vulnerability

New alerts

Security Alerts:

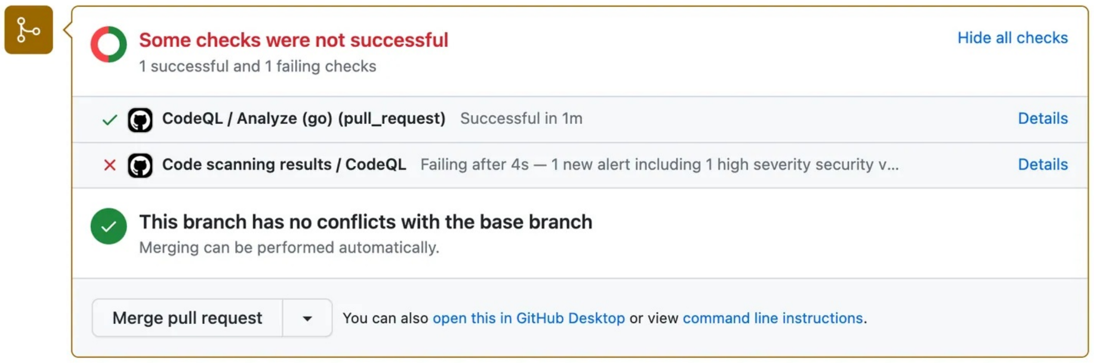
- 1 high

See annotations below for details.

Check failure on line 264 in index.js

Code scanning results check failures

If the code scanning results check finds any problems with a severity of `error`, `critical`, or `high`, the check fails and the error is reported in the check results. If all the results found by code scanning have lower severities, the alerts are treated as warnings or notes and the check succeeds.



Some checks were not successful
1 successful and 1 failing checks

CodeQL / Analyze (go) (pull_request) Successful in 1m

Code scanning results / CodeQL Failing after 4s — 1 new alert including 1 high severity security v...

This branch has no conflicts with the base branch
Merging can be performed automatically.

You can also open this in GitHub Desktop or view command line instructions.

You can override the default behavior in your repository settings, by specifying the level of severities and security severities that will cause a pull request check failure. For more information, see "Customizing your advanced setup for code scanning".

Other code scanning checks

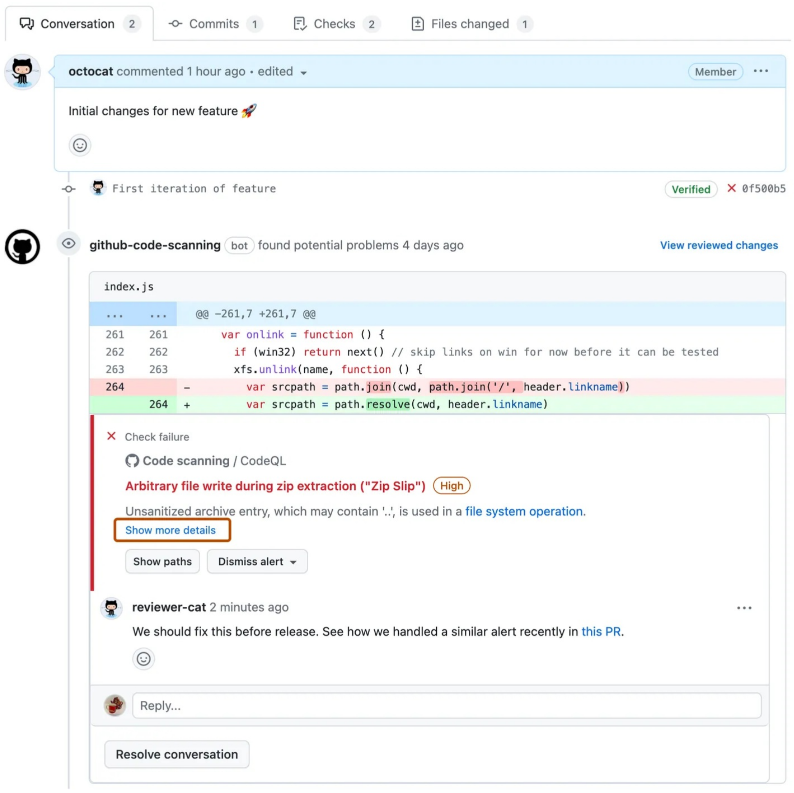
Depending on your configuration, you may see additional checks running on pull requests with code scanning configured. These are usually workflows that analyze the code or that upload code scanning results. These checks are useful for troubleshooting when there are problems with the analysis.

For example, if the repository uses the CodeQL analysis workflow a **CodeQL / Analyze (LANGUAGE)** check is run for each language before the results check runs. The analysis check may fail if there are configuration problems, or if the pull request breaks the build for a language that the analysis needs to compile (for example, C/C++, C#, or Java).

As with other pull request checks, you can see full details of the check failure on the **Checks** tab. For more information about configuring and troubleshooting, see "Customizing your advanced setup for code scanning" or "Troubleshooting code scanning."

Viewing an alert on your pull request

You can see any code scanning alerts that are inside the diff of the changes introduced in a pull request by viewing the **Conversation** tab. Code scanning posts a pull request review that shows each alert as an annotation on the lines of code that triggered the alert. You can comment on the alerts, dismiss the alerts, and view paths for the alerts, directly from the annotations. You can view the full details of an alert by clicking the "Show more details" link, which will take you to the alert details page.



octocat commented 1 hour ago • edited

Initial changes for new feature 🚀

First iteration of feature ✗ 0f500b5

github-code-scanning bot found potential problems 4 days ago

index.js

```
...    ...      @@ -261,7 +261,7 @@
261    261          var onlink = function () {
262    262            if (win32) return next() // skip links on win for now before it can be tested
263    263            xfs.unlink(name, function () {
264         -           var srcpath = path.join(cwd, path.join('/', header.linkname))
       264  +           var srcpath = path.resolve(cwd, header.linkname)
```

✗ Check failure

Code scanning / CodeQL

Arbitrary file write during zip extraction ("Zip Slip") High

Unsanitized archive entry, which may contain '..', is used in a file system operation.

reviewer-cat 2 minutes ago

We should fix this before release. See how we handled a similar alert recently in this PR.

You can also view all code scanning alerts that are inside the diff of the changes introduced in the pull request in the **Files changed** tab.

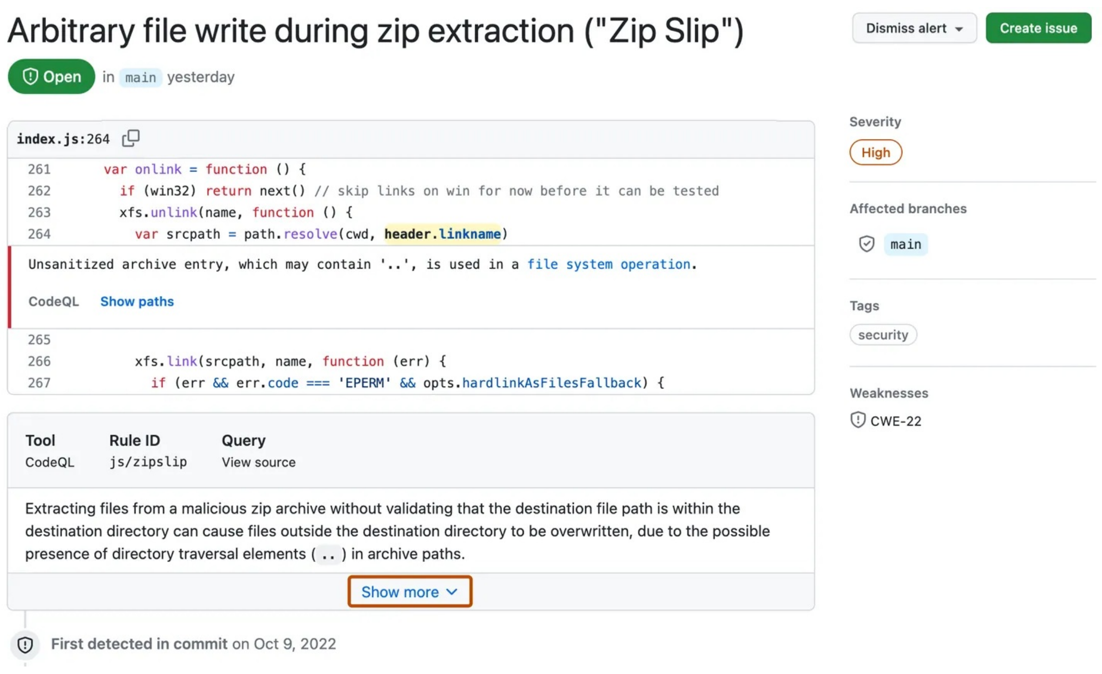
If you add a new code scanning configuration in your pull request, you will see a comment on your pull request directing you to the **Security** tab of the repository so you can view all the alerts on the pull request branch. For more information about viewing the alerts for a repository, see "Managing code scanning alerts for your repository."

If you have write permission for the repository, some annotations contain links with extra context for the alert. In the example above, from CodeQL analysis, you can click **user-provided value** to see where the untrusted data enters the data flow (this is referred to as the source). In this case you can also view the full path from the source to the code that uses the data (the sink) by clicking **Show paths**. This makes it easy to check whether the data is untrusted or if the analysis failed to recognize a data sanitization step between the source and the sink. For information about analyzing data flow using CodeQL, see "About data flow analysis."

To see more information about an alert, users with write permission can click the **Show more details** link shown in the annotation. This allows you to see all of the context and metadata provided by the tool in an alert view. In the example below, you can see tags showing the severity, type, and relevant common weakness enumerations (CWEs) for the problem. The view also shows which commit introduced the problem.

The status and details on the alert page only reflect the state of the alert on the default branch of the repository, even if the alert exists in other branches. You can see the status of the alert on non-default branches in the **Affected branches** section on the right-hand side of the alert page. If an alert doesn't exist in the default branch, the status of the alert will display as "in pull request" or "in branch" and will be colored grey.

In the detailed view for an alert, some code scanning tools, like CodeQL analysis, also include a description of the problem and a **Show more** link for guidance on how to fix your code.



Arbitrary file write during zip extraction ("Zip Slip")

index.js:264

```
    var onlink = function () {
      if (win32) return next() // skip links on win for now before it can be tested
      xfs.unlink(name, function () {
        var srcpath = path.resolve(cwd, header.linkname)
```

Unsanitized archive entry, which may contain '..', is used in a file system operation.

```

        xfs.link(srcpath, name, function (err) {
          if (err && err.code === 'EPERM' && opts.hardlinkAsFilesFallback) {
```

Tool	Rule ID	Query
CodeQL	js/zipslip	View source

Extracting files from a malicious zip archive without validating that the destination file path is within the destination directory can cause files outside the destination directory to be overwritten, due to the possible presence of directory traversal elements (..) in archive paths.

Commenting on an alert in a pull request

You can comment on any code scanning alert that appears in a pull request. Alerts appear as annotations in the **Conversation** tab of a pull request, as part of a pull request review, and also are shown in the **Files changed** tab.

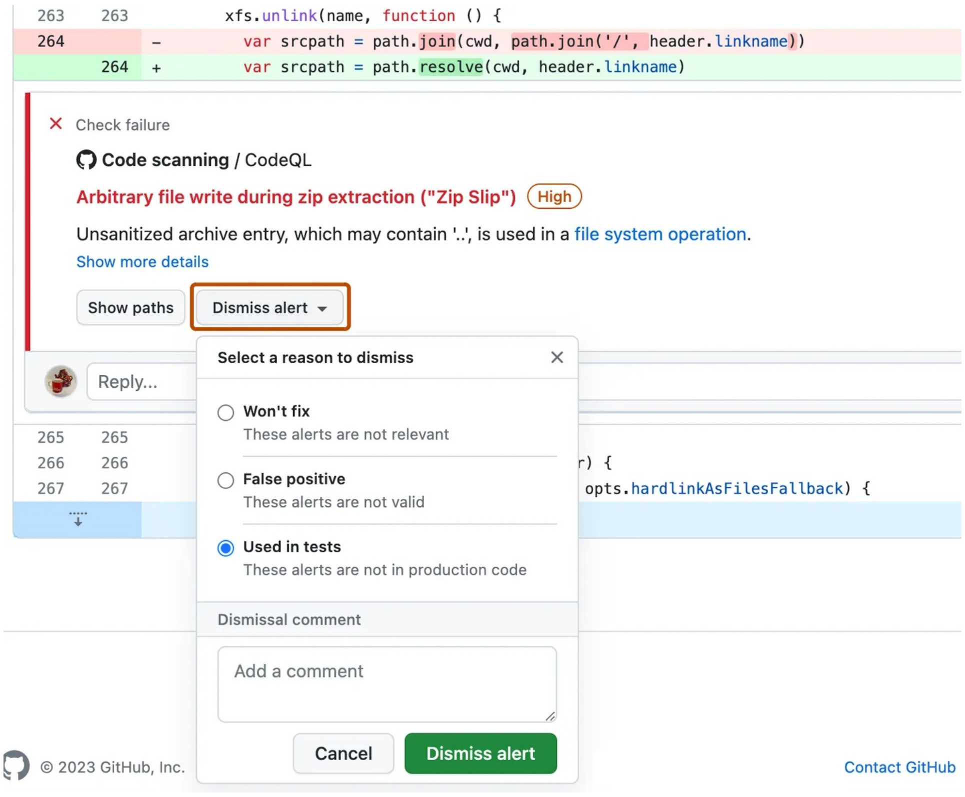
You can choose to require all conversations in a pull request, including those on code scanning alerts, to be resolved before a pull request can be merged. For more information, see "About protected branches."

Fixing an alert on your pull request

Anyone with push access to a pull request can fix a code scanning alert that's identified on that pull request. If you commit changes to the pull request this triggers a new run of the pull request checks. If your changes fix the problem, the alert is closed and the annotation removed.

Dismissing an alert on your pull request

An alternative way of closing an alert is to dismiss it. You can dismiss an alert if you don't think it needs to be fixed. For example, an error in code that's used only for testing, or when the effort of fixing the error is greater than the potential benefit of improving the code. If you have write permission for the repository, a **Dismiss alert** button is available in code annotations and in the alerts summary. When you click **Dismiss alert** you will be prompted to choose a reason for closing the alert.



```
        xfs.unlink(name, function () {
-          var srcpath = path.join(cwd, path.join('/', header.linkname))
+          var srcpath = path.resolve(cwd, header.linkname)
```

✕ Check failure

Code scanning / CodeQL

Arbitrary file write during zip extraction ("Zip Slip") High

Unsanitized archive entry, which may contain '..', is used in a file system operation.

Show more details

```
r) {
opts.hardlinkAsFilesFallback) {
```

Select a reason to dismiss ✕

- Won't fix — These alerts are not relevant
- False positive — These alerts are not valid
- [✓ Used in tests] — These alerts are not in production code

Dismissal comment

It's important to choose the appropriate reason from the drop-down menu as this may affect whether a query continues to be included in future analysis. Optionally, you can comment on a dismissal to record the context of an alert dismissal. The dismissal comment is added to the alert timeline and can be used as justification during auditing and reporting. You can retrieve or set a comment by using the code scanning REST API. The comment is contained in `dismissed_comment` for the `alerts/{alert_number}` endpoint. For more information, see "Code Scanning."

If you dismiss a CodeQL alert as a false positive result, for example because the code uses a sanitization library that isn't supported, consider contributing to the CodeQL repository and improving the analysis. For more information about CodeQL, see "Contributing to CodeQL."

For more information about dismissing alerts, see "Managing code scanning alerts for your repository."