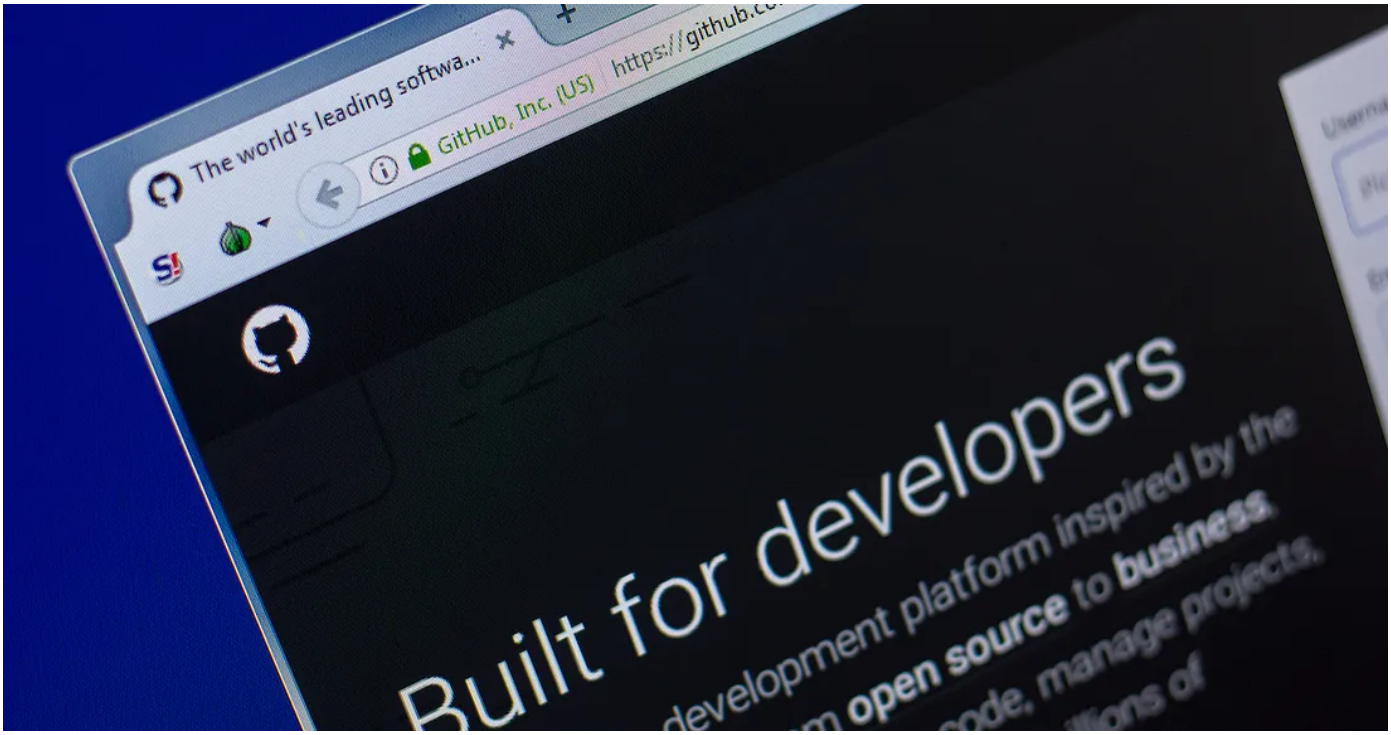
Omer Gil Follow

Oct 12, 2021 · 6 min read · Listen

Bypassing required reviews using GitHub Actions

Not using GitHub Actions? You're also vulnerable.



TL;DR

A newly discovered security flaw in GitHub allows leveraging GitHub Actions to bypass the required reviews mechanis ode to a protected branch,

potentially allowing malicious code to be used by other users or flow down the pipeline to production.

The risk of a compromised user account

GitHub is the most popular source control management system, serving millions of users and companies who use it to host their codebases. A GitHub organization can include any number of members — from several to hundreds or even thousands of members, with varying permissions. Write permissions are commonly granted to many users, as that is the base permission needed to directly push code to a repo.

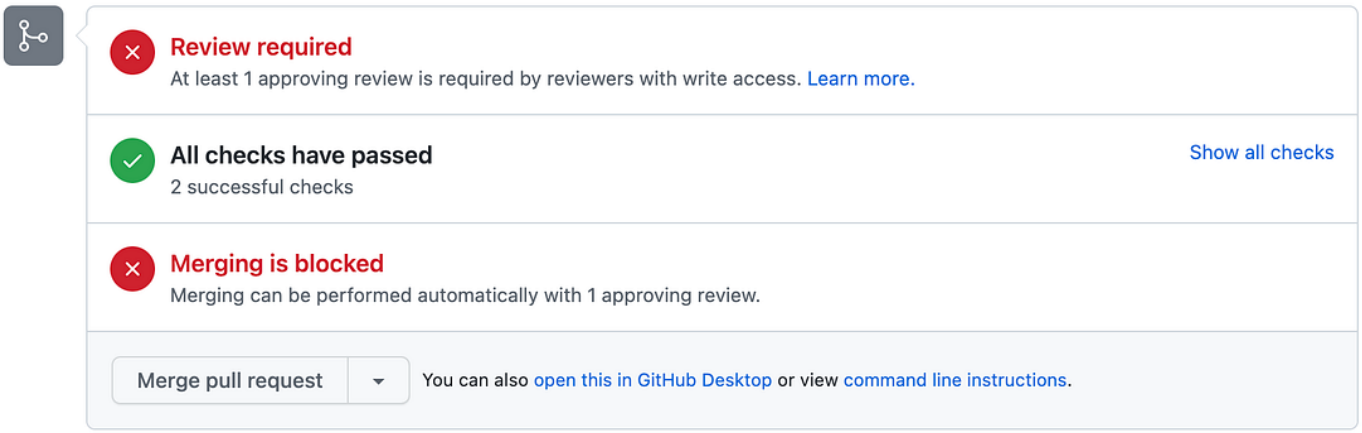
As GitHub organization owners are aware of the constant need to protect their code against different types of threats, one attack vector that is always of great concern is that of a compromised user account. Because if an attacker is able to take control of an account with Write permissions (by obtaining their password, personal access token, or an SSH key), they can directly push code to the repo, which might be used by other software and users. This code can also go down the CI/CD pipeline, run unreviewed in the CI, or find itself in the company's production environment.

So does a compromise of a single user account mean the attacker can push code down the pipeline without restrictions?

Required reviews for merge

To avoid this exact scenario (and for quality considerations, obviously), branch protection rules were created, and are used by nearly all engineering organizations today to provide baseline protection against such attack vectors. For sensitive branches (such as the default one — or any other branch we'd want to protect), we can set rules to limit an account with Write permissions to directly push code to it by requiring the user to create a pull request. Once a pull request is created, it needs to be approved by a preset number of approvers before it can be merged to the target branch.

For obvious reasons, a user cannot approve their own pull request, meaning that a requirement of even one approval, forces another organization member to approve the merge request in the codebase.



Creating these protection rules that require one approval on a pull request by another organization member significantly reduces the risk of compromising an account, as the code needs to be manually reviewed by another user[1]. This also prevents developers from pushing unreviewed code to sensitive branches.

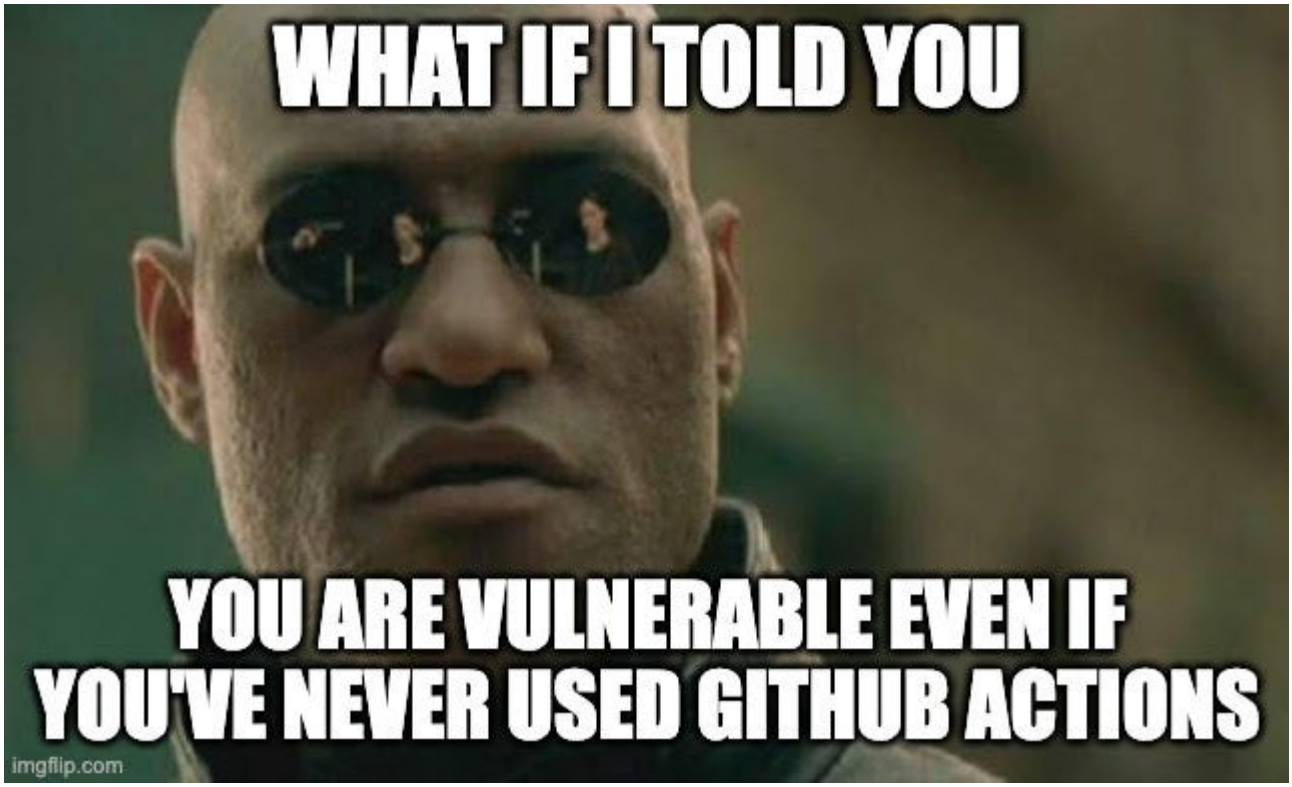
GitHub Actions

GitHub has evolved significantly since its inception — and continues to add features, products, and tools for code management and shipment. One such tool is GitHub Actions — GitHub's CI service — which is used to build, test, and deploy GitHub code by building and running workflows from development to production systems.

Why is GitHub Actions relevant to bypassing protected branch configurations?

Our research has exposed a flaw that leverages GitHub Actions to bypass protected branch restrictions reliant on the multiple reviews control.

So if your organization uses GitHub, but doesn't use GitHub Actions for CI, you obviously have no reason to be concerned about this flaw, right?



This begs the question, if you are an organization using GitHub, but haven't yet gotten started with GitHub Actions, should you be worried about GitHub Actions' attack surface, even if you never installed or used it in your organization? Let's see.

1. GitHub Actions is installed by default on any GitHub organization, and on all of its repositories.

2. Any user that can push code to the repo (Write permissions or higher), can create a workflow that runs when code is pushed.

3. With each workflow run, GitHub creates a unique GitHub token (GITHUB_TOKEN) to use in the workflow to authenticate against the repo. These permissions have a default setting, set in the organization or repository level. This setting allows granting the token with restricted permissions — *Read* permission on the contents and metadata scopes, or permissive permissions — *Read/Write* permissions on various scopes, such as contents, packages, and pull requests.

4. However...

> *Anyone with write access to a repository can modify the permissions granted to the GITHUB_TOKEN, adding or removing access as required, by editing the permissions key in*

> *the workflow file.*

Getting to the point

So, what does a typical GitHub organization look like?
It generally has:

1. Branch protection rules that can be set by organization owners to require pull request approvals before merge, where a user cannot approve their own pull request.
2. GitHub Actions installed by default for all GitHub organizations, on all repositories.
3. Permission for any user with Write access to run a workflow in the repo.

Practically, this means an attacker that hijacks a user account and wants to push code to a protected branch, can simply push their malicious code to a new remote branch, along with a workflow with the following content:

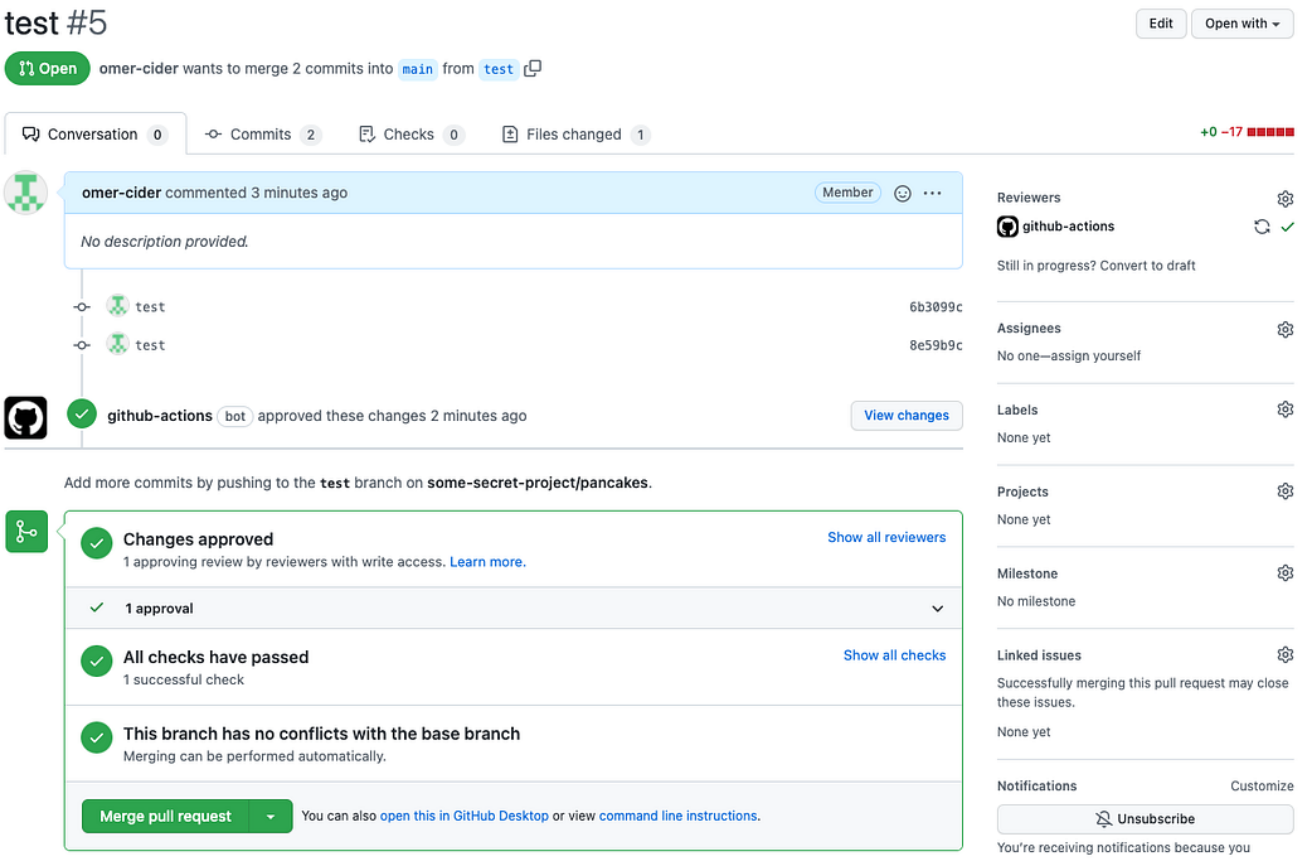
- Workflow code is aimed to approve the PR using the GitHub API.
- Workflow is configured to run on *pull_request* events. That includes creations and updates of PRs.
- Workflow is granted with Write permissions on the pull requests API endpoint.

Workflow code to automatically approve a PR

Then, the attacker creates a pull request, with the intent to merge their malicious code to a protected branch. As the PR is created, it cannot be merged since approval is required. However, the workflow immediately runs and the PR is approved by the *github-actions* bot, which the GITHUB_TOKEN belongs to.

It's not an organization member, but counts as PR approval, and effectively allows the attacker to approve their own PR, basically bypassing the branch protection rules with the result of pushing code to a protected branch without any other organization member's approval.

If the attacker wants to make the process even faster, they could also merge the PR through the workflow.



The PR is successfully approved with "github-actions" as the reviewer, and the code can be merged

PoC video

Any organization using GitHub as its codebase repository, trusting the security mechanism of required reviews to protect against direct push of code to sensitive branches, actually lacks this protection by default, even if GitHub Actions was never installed or used in the organization.

Report to GitHub

This security issue was reported to GitHub through their bug bounty program. They accepted it, wrote that it'll be tracked internally until resolved, and approved to publish a write-up.

Timeline

15/09: Reported to GitHub bug bounty program
15/09 : First response from GitHub
22/09: Triage
22/09: Payout
23/09: Approval for write-up

Mitigation

- If you're not using GitHub Actions, disable it for the entire organization or for specific repositories where it's not required.

- If GitHub Actions is in use in the organization, you can do one of the following:
  - Require a review approval in pull requests from Code Owners.
  - Increase the required number of approvals to 2 or more.
  - Wait until the issue is resolved by GitHub.

Update: January 2022

Following this blog post, GitHub recently introduced a new setting to fix this vulnerability. Organization admins can now disallow GitHub Actions from approving pull requests.

This is an organization-wide setting, which by default allows Actions to approve pull requests in existing organizations, and disallows it in newly created orgs. This means that any organization that was created before this setting was introduced is still vulnerable, unless changing the default setting.

To disallow Actions from approving pull requests, browse to Actions under Organization Settings. Look for this setting:

Clearing this setting will prevent Actions from approving PRs. The text is a bit misleading, as it's explained like Actions can approve a pull request — and it just won't count as an approval for merge, while practically it prevents approvals entirely.

```
1  ▶ Run curl --request POST \
8  {
9    "message": "Unprocessable Entity",
10   "errors": [
11     "GitHub Actions is not permitted to approve pull requests."
12   ],
```

We recommend you to use this new setting to disallow malicious actors from bypassing branch protection rules by approving their own pull requests.

Kudos to GitHub for fixing this security flaw.

[1] Obviously no one guarantees the approver actually reads the code, but at least now there's who to blame, right?

Github Github Actions DevOps Bug Bounty Appsec