

Product Releases | JANUARY 14, 2025 | 7 MIN READ

December 2024 Unauthorized Kong Ingress Controller 3.4.0 Build



Saju Pillai
SVP of Engineering, Kong



Wanny Morellato
VP, Security, Kong



Andrew Jessup
Principal Product Manager, KIC, Kong



Charly Molter
Senior Engineering Manager, KIC, Kong

Summary

On January 2, 2025, Kong became aware that an unauthorized actor had been able to publish a Docker image containing a cryptominer to the Kong DockerHub registry that affected v. 3.4.0 of the open source Kong Ingress Controller (KIC).

Our security team immediately revoked any public access to the affected image, rotated all secrets used in the repository, revoked the DockerHub access tokens, and rebuilt and re-published the correct image from a sterile build environment. We then began the process of performing initial analysis to identify the blast radius of the attack, notifying affected customers and publishing a community security advisory on GitHub.

We also began a root cause analysis to understand how the compromise occurred, and began efforts to confirm the impact of the compromised image with the assistance of a third-party forensics firm.

We have determined the means of attack to be a “Pwn Request” exploit that was enabled through a Dependabot actor confusion attack. The exploit was triggered by a GitHub pull request on an unused branch in the open source KIC GitHub repository, which allowed the attacker to steal secrets associated with the KIC build pipeline. This allowed them to craft and publish an unauthorized build of KIC to DockerHub.

The unauthorized image was downloaded 202 times. The impact of this exploit was limited by nature of the payload (cryptomining), the scope (a single recent image was affected) and timing (over the relatively quiet holiday and New Year period).

Open source users who are concerned they may be affected are encouraged to follow the details in [our GitHub security advisory](#) [🔗](#) for guidance on how to identify if they are affected and how to remediate their environment if so.


The rest of this post shares some details of the exploit and outlines some of the steps we are taking with our open source projects to protect against similar exploits in the future. We note that our proprietary products have different build processes with additional security controls in place.

Background

To understand this exploit, it is necessary to understand a little about building continuous integration (CI) pipelines using GitHub Actions and their security.

GitHub Actions has emerged as a popular CI tool, particularly for open source projects

as it allows an easy mechanism for third-party contributions to be run under the same build and test processes as the project maintainers. A common workflow is for a contributor to create a separate “downstream” fork of a repository, make changes to that fork, and then request that the changes be incorporated into the original “upstream” repository through a pull request. When a pull request is created, a GitHub Actions pipeline may be triggered that performs activities such as building the contributor-modified code and running tests against it automatically.

This workflow simplifies life for both contributors and maintainers. But a challenge arises when it is necessary for the code in Actions of a pipeline to call out and authenticate to third-party systems. A best practice is for the credentials used to authenticate to be stored as [GitHub secrets](#)  that are securely stored and managed by the upstream project owner using GitHub tools. Any privileged actions that require access to these secrets must therefore be performed in the context of the upstream repository.

It is important to control who is allowed to perform any privileged actions, otherwise a third party might modify the actions in their fork to read out secrets from the upstream context and use them elsewhere, or perform other malicious actions.

To mitigate such risks, open source projects can adopt various strategies to restrict pipeline execution and control access to secrets. These strategies involve trade-offs between security and ease of contribution, making it important to strike the right balance to protect both maintainers and contributors.

One control GitHub provides is different triggering events that initiate a workflow either in the downstream contributor's fork (the `pull_request` event) or the original upstream context (the `pull_request_target` event). Initially, all workflows in the KIC repository intended to be run by contributors were triggered by the `pull_request` event, meaning the workflows could not access upstream secrets.

Timeline

August 28, 2024 - Initial misconfiguration and remediation

On August 28, 2024, we changed the triggering event for a single KIC workflow from `pull_request` to `pull_request_target` with additional permission checks to allow authorized community contributors, including the built-in GitHub application Dependabot, to run tests before raising PRs back to upstream. By allowlisting Dependabot for our privileged workflows, we could more quickly verify and then

Dependabot for our privileged workflows, we could more quickly verify and then incorporate Dependabot's proposed updates.

On November 25, 2024, we received an anonymous report from a security researcher that the KIC repository was susceptible to a “[Pwn Request](#)” exploit enabled through a [Dependabot actor confusion attack](#). Out of caution we rotated all secrets in the repository and reverted the affected workflow by switching the trigger event back to `pull_request`.

However, a loophole remained. While the fix was applied to the master branch and all branches under active development, we failed to apply the fix to some old unused branches. As a result, branches with workflows triggered by the `pull_request_target` remained in the KIC repository.

December 23, 2024 - Attacker reconnaissance

Our analysis of the GitHub audit logs shows that at 3:09 AM UTC on December 23, 2024, a user created a pull request from their fork of the KIC repository against the upstream. The PR included a commit that triggered the “Cacheract” tool to be downloaded from a Gist hosted by the same user.

Crucially, the PR targeted an old branch of the KIC repository which still had the `pull_request_target` vulnerability. Because of this the PR was able to bypass the protections that we put in place in November.

At this point the attacker had a mechanism to access secrets associated with the upstream repository. One of these was a GitHub personal access token used by a number of KIC automated processes for CI that had the ability to trigger other workflows on the KIC repository, as well as create and delete pull requests against the repository itself. All activity performed by this access token was captured in the GitHub audit logs.

December 24, 2024 - Unauthorized image published

Approximately 22 hours later the attacker launched a similar attack against the KIC repository. This time they targeted a branch that should have been protected from the Dependabot actor confusion attack; however, since they now had access to our GitHub PAT they had the ability to trigger protected workflows, as well as delete workflow logs to cover their tracks. This was confirmed by the GitHub audit logs.

Shortly after this the attacker made a tag push to the `docker.io/kong/kubernetes-ingress-controller` image on DockerHub, and then



used the PAT to delete a number of workflow logs. This leads us to believe that the malicious payload was inserted through a transiently compromised GitHub Action pipeline rather than a leaked access token to DockerHub itself. The `:latest`, `:3.4` and `:3.4.0` tags in DockerHub were now pointing to the compromised image.

On December 29, 2024, four users noted in a GitHub issue that running KIC v. 3.4.0 was leading to abnormally high CPU loads.

January 2, 2025 - Confirmation and response

On January 2, 2025, in an isolated lab environment we were able to confirm a process in the KIC 3.4.0 container image hosted was logging spurious messages and was making DNS requests to subdomains of `supportxmr.com` — a Monero cryptomining site. We identified that the behavior was present in the public image on DockerHub, but could not be reproduced in re-builds of the same image. We quickly identified that an unauthorized push had been made to our DockerHub repository on December 23.

We immediately revoked all DockerHub tokens that had the ability to push to our public repositories, suspending automated release pipelines. We also immediately deleted the `:3.4.0` image tag on DockerHub that referenced the unauthorized binary, to prevent further downloads of the compromised image. We then pushed a new correct build under a new tag `:3.4.1`, as well as the `:3.4` and `:latest` tags. We also rotated all secrets associated with the affected GitHub repository, and actively reviewed our GitHub and DockerHub audit logs to ensure no other images had been similarly compromised.

The following working day we published [a public GitHub advisory](#)  that explained how to identify and remediate affected deployments. A few days later we [submitted YARA rules](#)  to help security teams quickly identify potentially affected images.

At the same time we began an investigation in conjunction with a third-party forensics firm to analyze the unauthorized image. The analysis concluded that the only malicious payload was the XMRig cryptominer.

Looking forward

Our security and engineering teams have been working together to implement an enhanced set of build and release controls for our open source projects, including the following:

- Updating `workflow_permissions` on all of Kong's public repositories to read-only.

- Verifying that no other public repositories had reachable workflows triggered by `pull_request_target` events.
- Manual verification & lockdown of necessary `pull_request_target` workflows (such as for cherry-picking)
- Manual verification using scanning tools of Actions workflows for exploitability against Dependabot actor confusion.
- Auditing of the scope and necessity of all DockerHub push access tokens used at Kong.
- Eliminating cache reuse for deployment workflows to mitigate GitHub cache poisoning attacks.
- De-coupling build and deploy workflows, especially for formal releases, and separating privileged credentials to deploy images from those required to execute builds with a “human air gap”.
- Running build artifacts in an isolated environment for a period of time to test for anomalous behavior, before releasing.
- Migrating from static access tokens to Workload Identity Federation.
- Re-evaluating our incident response protocols (IRP) for open source projects.

Some of these changes we have made already, and some we will be implementing in the near future.

Conclusion

While it was fortunate that the exploit was limited to a relatively small number of affected users, it has served as a reminder of the necessity of defense-in-depth when providing convenience to the open source community.