Ultralytics AI Library Hacked via GitHub for Cryptomining

By Wiz Threat Research

Published: 2024-12-09 · Archived: 2026-04-05 17:26:44 UTC

Security researchers have identified a supply chain attack targeting deployment versions of the Ultralytics Python package. The compromised versions, `8.3.41` and `8.3.42`, contain malicious code that executes unauthorized cryptocurrency mining software (XMRig) on affected machines. This compromise was limited to the PyPI-hosted versions of the package, and local or earlier versions remain unaffected. The malicious versions have since been removed from PyPI to prevent further exploitation. Ultralytics is a popular AI image prediction library with over 33k stars on GitHub and a dependency for many packages, including the popular ComfyUI Impact Pack extension, making them vulnerable as well.

Why is this interesting?

PyPI-hosted package compromise is nothing new. What is interesting in this case is the way the package was compromised – via the CI/CD workflow. Until recently, most compromises related to VCS (version control system)- and CI/CD were performed by security researchers (i.e. on stripe-samples, PyTorch, GitHub Runner Images, etc.).

Even the infamous XZ-Utils compromise was performed by what could be considered an “insider threat”, someone that already had write access to the repository code. By contrast, this is one of the first publicly known cases where an external malicious actor managed to modify the release package that eventually made it to the PyPI ecosystem. It is also notable that the discovery of the compromise was first alerted on the dependent package ComfyUI, not on the original compromised target Ultralytics.

What happened?

A supply chain attack targeted Ultralytics, a very popular library which is included in many AI packages including the ComfyUI Impact Pack. The attacker manipulated GitHub Actions by exploiting branch names in pull requests to execute arbitrary code, bundling a cryptominer into the package. A subsequent “mitigation” release of Ultralytics was also compromised, escalating the risk for users who might have updated to the new version and assumed they were secure.

How exactly did this happen?

Ultralytics has an extensive CI/CD infrastructure boasting 11 different workflows running tens of job runs every hour. This probably contributed to the fact that the malicious change was not immediately noticed and that two versions of the Ultralytics package were shipped successfully and made it to the PyPI registry.

On December 4th, 2024, a GitHub user named *openimbot* opened two strange-looking draft pull requests . The purpose of these PRs was to exploit a vulnerability that was similar to one previously reported as affecting a different workflow in the same organization (Ultralytics) but in different repository (Ultralytics/actions). At the

center of the malicious PRs was the vulnerable workflow “Publish Docs” that runs upon every PR creation (among other triggers):

Specifically, the vulnerable lines of code are best seen when analyzing the later fix by the package maintainers:

In this case, `github.head_ref` is the name of the source code branch opened with the PR. When treating `github.head_ref` as a string and without input sanitization, the workflow used the maliciously crafted branch name supplied by *openimbot* in the above PRs:

This branch name payload, when executed by the job, pipes the content of `file.sh` into a bash session. The way the payload is constructed (using the parentheses bash notation and the `$IFS` special shell variable) is explained by the fact that, to be valid, the branch name should not contain any spaces. In fact, this is a common technique used for WAF bypasses. The content of `file.sh` is currently unavailable, but we can speculate that it contained instructions to check in malicious changes to the following two files:

- `v8.3.41/ultralytics/models/yolo/model.py` - Adding code that downloads and runs a malicious miner according to the victim’s machine:
- `v8.3.41/ultralytics/utils/downloads.py` - Adding the implementation of the download and run functions used in `model.py` :

The final result is evident in this issue raised by a user of Ultralytics that noticed the discrepancy.

It is somewhat surprising to see these malicious actions performed by an established GitHub account with a long history of contributions. After all, opening a temporary GitHub account is something very common and easy achievable, whereas creating and maintaining a believable persona over a long period of time is slightly more difficult. However, it is quite possible that this account is legitimate and was compromised somehow by a malicious actor in order to enact supply chain attacks such as this.

Wiz Research data: what’s the risk to cloud environments?

Based on our data, Ultralytics itself can be found in 10% of cloud environments, demonstrating the valuable attack surface that this supply chain attack was aiming to exploit.

Which products are affected?

Ultralytics in versions `8.3.41` and `8.3.42` contain malicious code. Later versions are safe to use.

Which actions should security teams take?

Users who have installed these versions are strongly advised to uninstall the package immediately and restore impacted systems to a previously known clean state, while monitoring for any evidence of crypto-mining on the affected systems.

Wiz customers can use the pre-built query and advisory in the Wiz Threat Intel Center to search for affected instances in their environment:

References

- kingbri's tweet
- Github issue for Ultralytics
- Github issue for ComfyUI
- ReversingLabs blog
- BleepingComputer article
