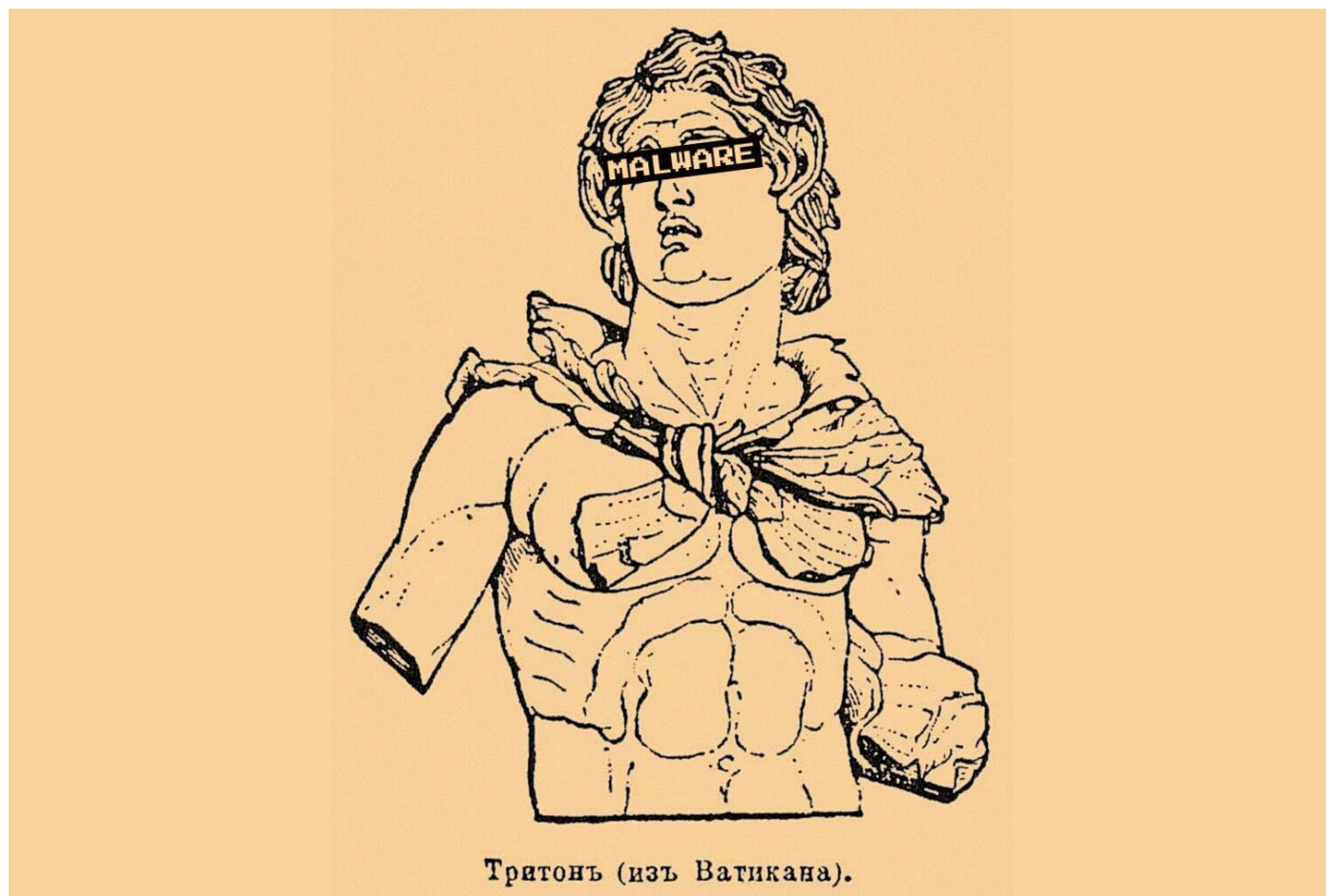


The Curious Case of the Triton Malware Fork · brennan.day

 brennan.day/the-curious-case-of-the-triton-malware-fork

Brennan Kenneth Brown

February 15, 2026



Triton Illustration from Brockhaus and Efron Encyclopedic Dictionary (1890—1907) | [Source](#) (Edited by the Author)

Earlier today, I was in [omg.lol's](#) IRC server when discussion arose about a fork of a lovely MacOS app for omg.lol called [Triton](#). Somebody seemed to have copied the repository's source code wholesale and removed any mention of the original creator.

At first, it seemed as though this fork was a case of plagiarism. I'm not a stranger to this. During my time as an open-source developer, I've searched up specific code snippets unique to my codebases and have found people passing my work off as their own for school assignments. My README [is a popular example](#), for whatever reason.

This was not the case this time.

Ethical vs. Unethical vs. Illegal

It's important to start off by defining the difference between building off someone else's work ethically vs. unethically, since these are open-source projects, usually with very permissible licenses.

When you use someone's open-source code, there's what's *legal*, what's *ethical*, and what's just straight-up *illegal*.

As [I've written about recently](#), a lot of developers use the [MIT license](#), meaning that people can essentially do whatever they want with your work. [The MIT license is permissive](#) and allows users to incorporate MIT-licensed code into proprietary (commercial) software without releasing their source code. The [only legal requirement is attribution](#), so you must include the original copyright notice and the license text in your copy or derivative work. That's the bare minimum.

But [just because something is legal doesn't mean it isn't shitty](#). The open-source community operates on norms that go beyond legal compliance. [Ethical behaviour is to give proper recognition and attribution to all authors and contributors](#) instead of burying a copyright notice in a LICENSE file and calling it a day.

Ethical: You fork the project, keep the original author's name in the README, write "Based on [Original Project] by [Author]" prominently and link back to the original repo. You're transparent about where the code came from and respect the [spirit of software attribution](#).

Legal but unethical: You technically include the copyright notice somewhere deep in your codebase, meeting the legal requirement. But you strip the original author's name out of the README and remove all acknowledgments. You present the project as if it's entirely your own creation. Technically legal and morally bankrupt? The letter of the law is followed while violating the norms of the open-source community.

Illegal: You don't include the copyright notice or license text at all. You just take the code and pretend it's yours. This violates the license terms and is actual copyright infringement.

After actually having a look at the repository myself, I realized rather quickly this was not someone wanting to take Triton and pass off the hard work of [Otávio C.](#) as their own.

No, this was somebody who was trying to get people to download malware. And they did so in many different, interestingly stupid ways.

The [README.md](#)

To start, the README is bizarre, to say the least. It's actually pretty funny, here are a few excerpts:

🐙 Triton - Your Easy Solution for
https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip

🚀 Getting Started

Welcome to Triton! This is a native macOS client for

https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip

With Triton, you can enjoy features like Statuslog, PURLs, Web Pages, Now Pages, Weblog, Pics, and Pastebin—all in one place.

📦 Download Triton

[![Download Triton]
(https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip%https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip)]
(https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip)

To get started, you need to download Triton.

Click the button above, or visit this page to download:

[Triton Releases]

(https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip).

In the middle of the README:

🤝 Contributing

We welcome contributions from everyone.

If you would like to help, check our [Contributing Guidelines]

(https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip).

And at the end of the README:

📦 Download Triton Again

For quick access, you can download Triton here:

[Download Triton]
(https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip).

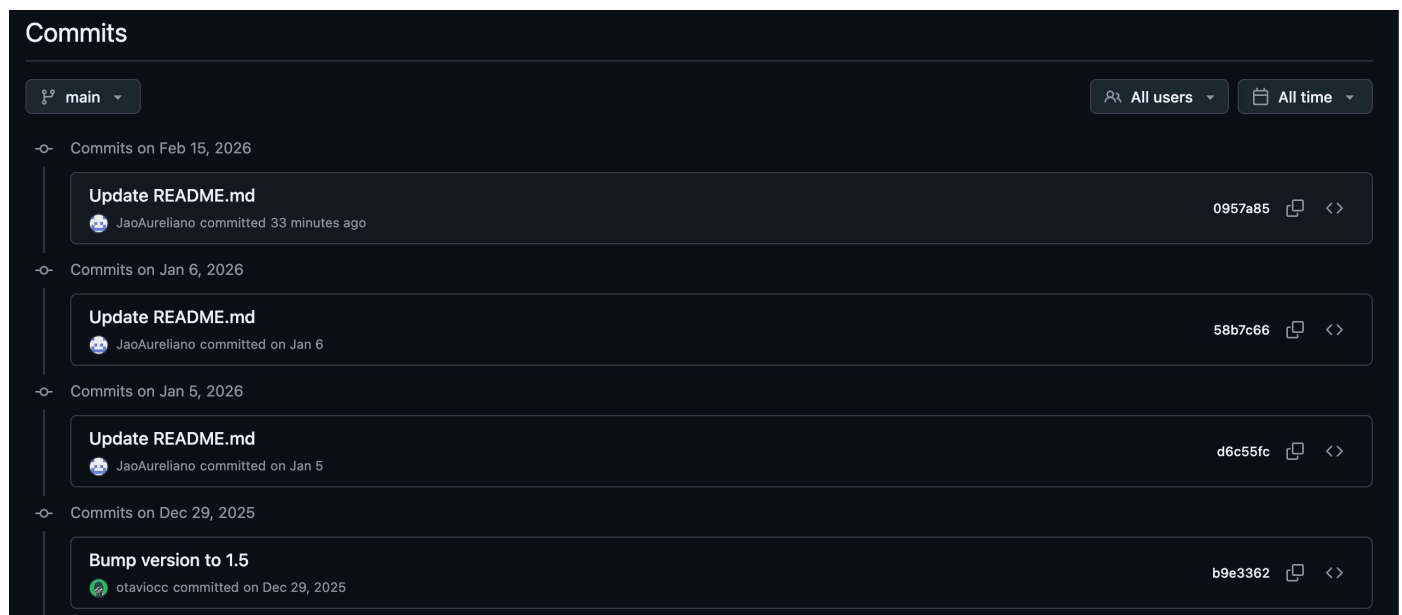
As you can see, there was absolutely no discretion here. They really just said "fuck it" and jammed the malicious link as many times as possible throughout this README.

And it almost seems clever that the malicious software would be tucked away in the `.xcassets` of the source code... except that this isn't a case of downloading the macOS app as-is and running it and suddenly you have malware on your machine. No, you are pushed to download *the .ZIP itself*, not the actual Triton app.

As for the .ZIP file itself, I was ~~stupid~~ curious enough to download and analyze it. [Here are the VirusTotal results](#). (I'll get more into the weeds of what this program is later in the post.)

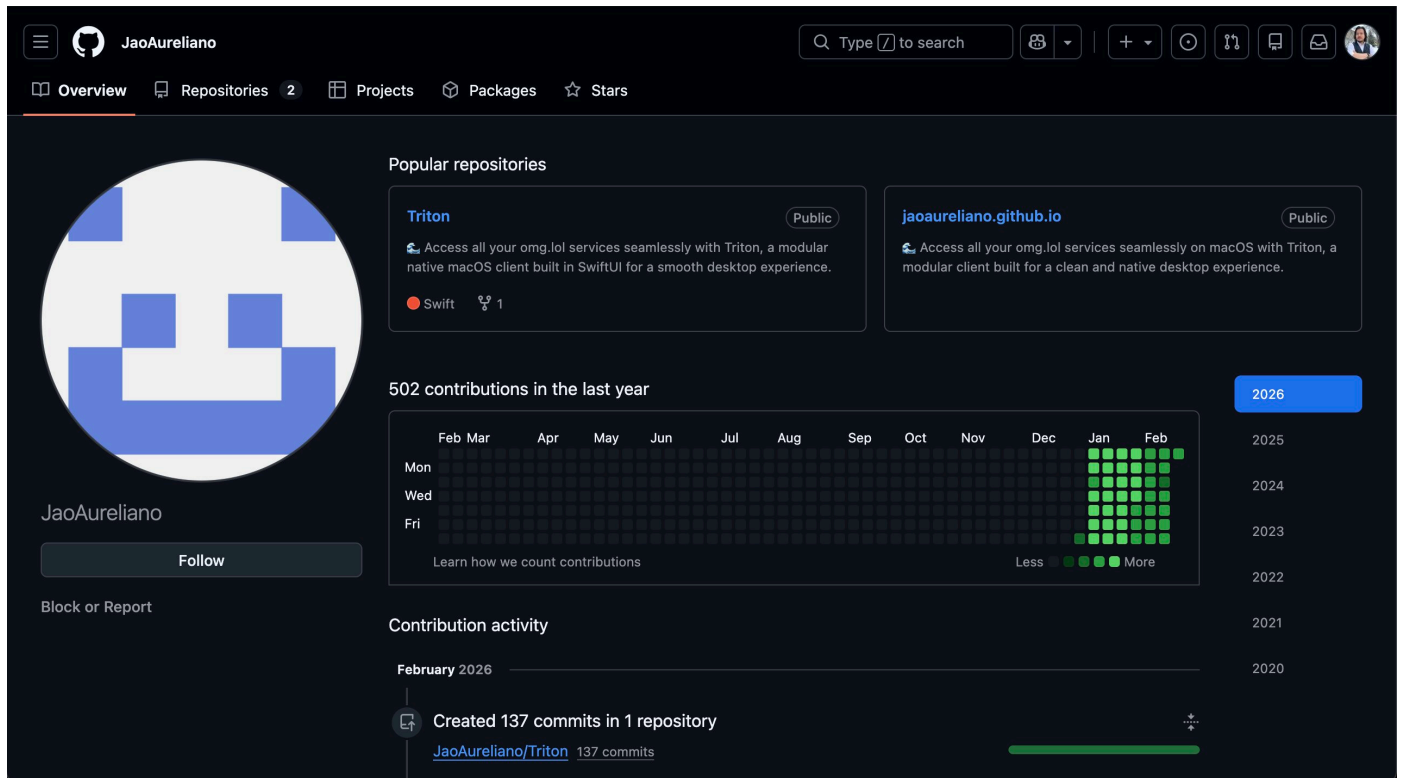
For now, I want to go over how bizarre this account is. There are only two repositories on the account, making this a rather targeted malware campaign attempt. The account is named "JaoAureliano" which would be a typical name in South America. Jao is a Portuguese diminutive of "João" (John) and Aureliano is derived from the Latin Aurelius, which means "golden" or "gilded."

Next, if you look at the commit history of the two repositories, they are rather sparse.



GitHub commit history showing sparse commits on the malicious repository

But if you look at the account's contribution graph, it has clearly been manipulated. It's clear that the commit history has been obfuscated by using an automated script to flood the graph and backdate dummy commits in an attempt to hide the malware injection.



GitHub profile showing manipulated contribution graph with backdated commits

What does the Malware do?

This is the funniest part, for me. The malware would only be able to trigger its payload on a Windows machine. In other words, a tiny open-source MacOS app was hijacked with a Windows virus.

Let me get in the weeds of it.

Attack Overview

Target: Triton (otaviocc/Triton) - A legitimate macOS application

Malicious Fork: JaoAureliano/Triton

Malware Hash: `39b29c38c03868854fb972e7b18f22c2c76520cfb6edf46ba5a5618f74943eac`

Detection Rate: 12/66 vendors (VirusTotal)

File: Software_3.1.zip (1.33 MB)

The attacker created a fork (JaoAureliano/Triton) with modifications:

- Download link redirected to:
`https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip`
- Malware hosted in an Xcode colorset directory, and poorly presented to end-user
- Repository topics, for some reason, include: `malware`, `deobfuscation`, `pytorch`, `symbolic-execution`, `llm-training`

Malware Analysis

Based on VirusTotal sandbox execution, the malware goes through the following stages:

Stage 1: Archive Extraction

Process: 7za.exe

Command: "C:\Windows\System32\7za.exe" x -pinfected -y -

o"C:\Users\user\AppData\Local\Temp\tibe4tu5.xl4" "C:\Users\user\Desktop\Software_3.1.zip"

- Uses password "infected" (common in malware analysis circles)
- Extracts to a randomly-named temporary directory

Stage 2: Execution Chain

Process: cmd.exe

Command: "cmd.exe" /C "C:\Users\user\AppData\Local\Temp\tibe4tu5.xl4\Launcher.cmd"

- Batch script initiates payload deployment
- References [luajit.exe](#) (Lua Just-In-Time compiler) for scripting

Stage 3: Evasion Techniques

Anti-Analysis Features:

- [detect-debug-environment](#) - Checks for debugging tools
- [long-sleeps](#) - Delays execution to evade automated sandboxes
- Virtualization/Sandbox Evasion (T1497) - Multiple detection techniques

MITRE ATT&CK Tactics:

- TA0002 (Execution): Command and Scripting Interpreter (T1059)
- TA0005 (Defense Evasion):
 - Deobfuscate/Decode Files (T1140)
 - Virtualization/Sandbox Evasion (T1497)
 - Impair Defenses (T1562)
- TA0007 (Discovery):
 - System Information Discovery (T1082)
 - Query Registry (T1012)
 - Process Discovery (T1057)
 - Virtualization/Sandbox Evasion (T1497)
- TA0011 (Command and Control):
 - Application Layer Protocol (T1071)
 - Encrypted Channel (T1573)
 - External Proxy (T1090)

Network Activity

DNS Resolutions:

nexusrules.officeapps.live.com
svc.ha-teams.office.com
ip-api.com
polygon-rpc.com

HTTP Requests:

GET http://ip-api.com/json/ # External IP lookup
POST http://89.169.12.160/api/NTEsN2QsN2UsNTgsNWIsNjAsNjIsNjcsYyw30Sw= # Encoded C2 communication
POST https://polygon-rpc.com/ # Blockchain RPC communication

These domains suggest:

- Command & Control (C2) disguised as legitimate Office traffic
- External IP discovery for victim profiling
- Blockchain/cryptocurrency functionality
- Data exfiltration through trusted infrastructure
- Evasion of basic network monitoring

Memory Patterns:

- Domain: lua-jit.org
- URL: <https://lua-jit.org/> Use of LuaJIT for scripting and code execution and evasion.

File System Operations

Paths Accessed:

%WINDIR%\SysWOW64
%WINDIR%\System32
C:\Program Files (x86)\Common Files\Oracle\Java\
C:\ProgramData\Microsoft\Windows Security Health\Logs
C:\Users\user\.dotnet\tools\
C:\Users\user\AppData\Local\Programs\Python\Python313\

The malware performs system reconnaissance, checking for:

- Development environments (Java, Python, .NET)
- Security software logs
- System directories for potential privilege escalation

Registry Keys Opened:

HKEY_CURRENT_USER\Software\Microsoft\Windows NT\CurrentVersion
HKEY_CURRENT_USER\Control Panel\International
HKEY_CURRENT_USER\Software\Microsoft\ .NETFramework

Suggests gathering system configuration data and potentially establishing persistence.

Attack Sophistication

There is currently a low detection rate, with only 12/66 vendors, suggesting recent deployment or effective evasion.

The repository topics include **malware**, **deobfuscation**, and **symbolic-execution**, perhaps as an attempt to appear legitimate to security researchers who might dismiss it as educational content. Or perhaps incompetence.

Possibly, the attacker is an OpenClaw instance, or student/amateur (I'll get more into this later).

Indicators of Compromise (IOCs)

File Hashes

SHA256: 39b29c38c03868854fb972e7b18f22c2c76520cfb6edf46ba5a5618f74943eac
Size: 1.33 MB
Type: ZIP archive containing PE executables

Malicious URLs

https://raw.githubusercontent.com/JaoAureliano/Triton/main/OMG/Assets/Assets.xcassets/AccentColor.colorset/Software_3.1.zip

Network IOCs

nexusrules.officeapps.live.com
svc.ha-teams.office.com

Process Names

7za.exe
unarchiver.exe
Launcher.cmd
luajit.exe

File Paths

C:\Users\[user]\AppData\Local\Temp\[random].xl4\Launcher.cmd

Why?

Good question. It's obvious this was an incredible failure of a malware campaign. The userbase of Triton is incredibly small, and for this to work, the end user would need to download the source code on their Windows machine instead of their Mac.

And yet, the malware itself seems competent.

I have a hypothesis: this was created by an [OpenClaw](#) agent, and this might be only one instance of hundreds. This is what I think happened:

1. The bot was given instructions to create a new GitHub account (VPNs, burner phone numbers, and other assets would have to have been utilized).
2. Then, it was instructed to find open-source programs on GitHub to steal and inject with malware, as well as creating their own GitHub pages repo for distribution.
3. After, it was instructed to artificially erase its git history by mass backlogging commits.

If I'm right, then this might have been the canary in the coalmine, and things are actually going to get so much worse when someone figures out how to have their OpenClaw instance do this in a way that could actually successfully target people.

I've already written about OpenClaw in [a previous article](#), but let me add more findings. [OpenClaw's skill catalogue is full of malicious code](#). There's hundreds of malicious script plugins, downloaded thousands of times, that [packaged malware which exfiltrated crypto wallets, browser passwords, and cloud credentials](#).

For now though, in this particular instance, nobody is in danger. I don't have to write out security tips, here. Nobody would take a look at this repository and download the .ZIP and run it on their Windows machine. Hurray.

And, maybe instead it is actually this bad on purpose. Perhaps this was merely supposed to be an illustrative example of what a malware campaign would look like, and it was done by a person instead of a water-guzzling, hallucinatory, fancy auto-complete.

It's Time to Leave GitHub

Despite numerous reports, GitHub has not taken down the account or malware as of me writing this, which is hours later. It seems trivially easy to host harmful software on Microsoft's coding platform for a surprisingly long amount of time. In the response to my report ticket, GitHub replied that they "wanted to let me know that they've received my message" and that they "are experiencing high volume and therefore, I may experience longer than normal wait times."

This may be due to having over [75 outages and incidents in the past 90 days alone](#).

Now that we've had our fun looking into some random, shitty malware, it's time to actually write something useful and actionable. In truth, this is only one of many reasons you should be looking into hosting your code elsewhere if you're a programmer. GitHub and Microsoft have demonstrated a pattern of decisions that conflict with the values of the independent, ethical web. Here are the key concerns:

Loss of Independence and Declining Engineering Quality

In August 2025, [GitHub's CEO resigned and GitHub was completely absorbed into Microsoft's CoreAI division](#), ending operational independence. This happened at the same time as [mounting complaints about declining service quality](#).

The [Zig Software Foundation publicly quit GitHub](#), citing Microsoft's AI obsession as having "ruined the service." Critical infrastructure bugs like the [safe_sleep.sh issue that caused runners to hang indefinitely](#) went unaddressed for over a year while Microsoft prioritized AI features.

Copilot Training on Your Code Without Proper Attribution

GitHub Copilot has been [trained on billions of lines of code from public repositories](#) without respecting open-source license requirements. A [class-action lawsuit filed in November 2022](#) alleges that Copilot violates the terms of 11 popular open-source licenses (including MIT, GPL, and Apache) by:

- Removing copyright management information and attribution
- Suggesting code without including required license terms
- [Monetizing developers' work without permission](#)

Privacy and Security Concerns

[Microsoft Copilot leaked thousands of private GitHub repositories](#) from major companies including IBM, Google, and Microsoft itself. Over 16,000 organizations were affected, with access keys and security tokens exposed through caching mechanisms.

[GitHub's privacy policy is deliberately vague](#) regarding training data. While Business and Enterprise customers reportedly have protections, free and Pro tier users face [uncertainty about whether their code is used for training](#). [Users report that privacy settings have been mysteriously re-enabled](#) without consent.

Ignored User Feedback

[User requests to disable or block Copilot features have been the #1 and #2 most popular topics](#) on GitHub's community page for over 12 months—and remain completely unanswered. Users report that [disable buttons are ignored by the interface](#).

Pricing Controversies

In December 2025, GitHub [announced plans to charge \\$0.002/minute for self-hosted Actions runners](#), charging developers to run GitHub's software on their own hardware. The backlash was so intense that GitHub [postponed the change within 24 hours](#).

How to Migrate Your Code

You don't have to stay on a platform that doesn't align with your values. Here are ethical alternatives:

[Codeberg](#) (Recommended)

A [non-profit, community-driven platform](#) run by Codeberg e.V., a democratic German organization. It's funded entirely by donations. No investors and no ads.

Features:

- Free Git hosting for open source
- CI/CD via Woodpecker and Forgejo Actions
- Static site hosting with Codeberg Pages
- [Migration tools built-in](#)

[Self-Hosted Options](#)

- [Forgejo](#): The software powering Codeberg. Lightweight, runs on modest hardware (like Raspberry Pi), with a familiar GitHub-like interface.
- [Gitea](#): The original project Forgejo forked from. Still actively maintained, written in Go, easy to install.
- [GitLab](#): Full DevOps platform with comprehensive CI/CD. Can be self-hosted or used as a service.

Migration Steps

1. **Create accounts** on your chosen platform(s)
2. **Use built-in migration tools**: Most alternatives can import directly from GitHub, including issues, PRs, and wikis
3. **Update your README**: Add a notice about the migration with links to new locations
4. **Archive GitHub repos**: Mark them as read-only with clear redirection
5. **Update local remotes**: `git remote set-url origin <new-url>`

[Detailed migration guide for Codeberg](#)

Disclaimer

This analysis is provided for educational and defensive security purposes. The information should be used to improve security posture and awareness, not for malicious purposes. Always report security incidents through appropriate channels.

UPDATE

I've spent the last several hours investigating what I initially thought was a single malicious fork of a macOS app. It turns out to be part of [a massive, coordinated campaign with hundreds of active malicious repositories](#).

I am not the first one to discover this. This [has been ongoing](#) for [at least a year](#) and nothing has been done.

Please report this to GitHub.



Queer Métis independent author and web developer based in Mohkínstsis/Calgary, Alberta (Treaty 7 territory). Full-time writer on Medium and founder and lead developer at Berry House. Previously: Founder of Write Club. Forever an advocate of the IndieWeb.

Enjoy this content? Support my work and help me create more:

[Patreon Ko-fi Sponsor](#)

Get new posts delivered to your inbox via RSS. No spam or selling, ever.

Enter your email

[Powered by Buttondown.](#)

Comments

To comment, please sign in with your website:

Your website:

No comments yet. Be the first to share your thoughts!

Webmentions

5 Likes

6 Reposts

3 Replies

@brennan LOL indeed.

Ah! found a write up of the other one I remember seeing <https://brennan.day/the-curious-case-of-the-triton-malware-fork/> @brennan wrote it up. The Curious Case of the Triton Malware Fork

I was right. This is widespread. A cursory search of "malware" (because for whatever reason, these malicious repos are literally tagged with it) has revealed multiple repositories following the exact same pattern: <https://github.com/KUNDANIOS/TheCha86> <https://github.com/Wothan1...>

3 Mentions

- <https://adsecvn.com/ma-doc-triton-gia-mao-tan-cong-nguy-hiem-tu-github/>
- <https://cybersecuritynews.com/malicious-fork-of-legitimate-triton-app/>
- <https://gbhackers.com/triton-app-discovered-on-github/>

Related Posts
