

Graphalgo fake recruiter test campaign respawned

 reversinglabs.com/blog/graphalgo-campaign-respawned

Karlo Zanki

April 9, 2026

Graphalgo fake recruiter campaign returns

An attack targeting crypto developers has been respawned — with an

LLC and new techniques to hide malware.

 Graphalgo supply chain campaign respawned.

In February, the ReversingLabs research team described [a malicious campaign featuring fake job interviews](#) that the team called “graphalgo.” Two months later, RL researchers detected a larger set of fake companies that are part of the same graphalgo campaign — yet more sophisticated.

These organizations link to several GitHub organizations related to blockchain companies that have been active on GitHub since June 2025. Their purpose is to provide trustworthiness to fake job offerings, and to host fake job interview tasks.

RL researchers also identified several new techniques being used by threat actors. Here's what we found.

The new graphalgo campaign overview

As explained in their team's original blog post on graphalgo, the whole campaign can be split into several independent activities conducted by the threat actor. This modularity makes it easier for the threat actor to keep the campaign active even if some part of it becomes compromised or is taken down.

The original campaign featured front end phishing activities being conducted via job seeking platforms and social networks. Coding tasks posted by the job recruiter were used to lure developers into downloading and installing malware from a fake GitHub-based crypto organization: "veltrix-capital," with a separate web domain, [www.\[\]veltrixcap.\[\]org](http://www.[]veltrixcap.[]org), along with the GitHub account.

Those job tasks contained a dependency to a malicious package hosted on open source package repositories like npm or PyPI. That dependency would get executed during the setup procedure for the given job task. The sole purpose of the dependency was to download the final malicious payload: a RAT (remote access trojan) typical for this campaign.

The frontend operations

In the newly discovered campaign branches, those frontend phishing operations remain the same, but new GitHub organizations have been spawned to replace veltrix-capital, which was removed after the malicious campaign was exposed.

The newly discovered campaign includes two accounts that mimic [SWFT Blockchain, a legitimate company](#), and two fake companies: "Blockmerce" and "Bridgers Finance." A number of links to the Veltrix company — the central point of the initial frontend operations — have been observed in each one. That includes commits in the new organizations by authors with *veltrixcap.[]org* email addresses, veltrix-linked GitHub accounts following the new domains and more.

Like Veltrix Capital in the first campaign, Bridgers Finance is at the center of the ongoing activities. At the time of publication, a fake LinkedIn profile of a Bridgers "technical recruiter" named [Gnanika Thumba](#) is actively publishing job offering posts related to this campaign.

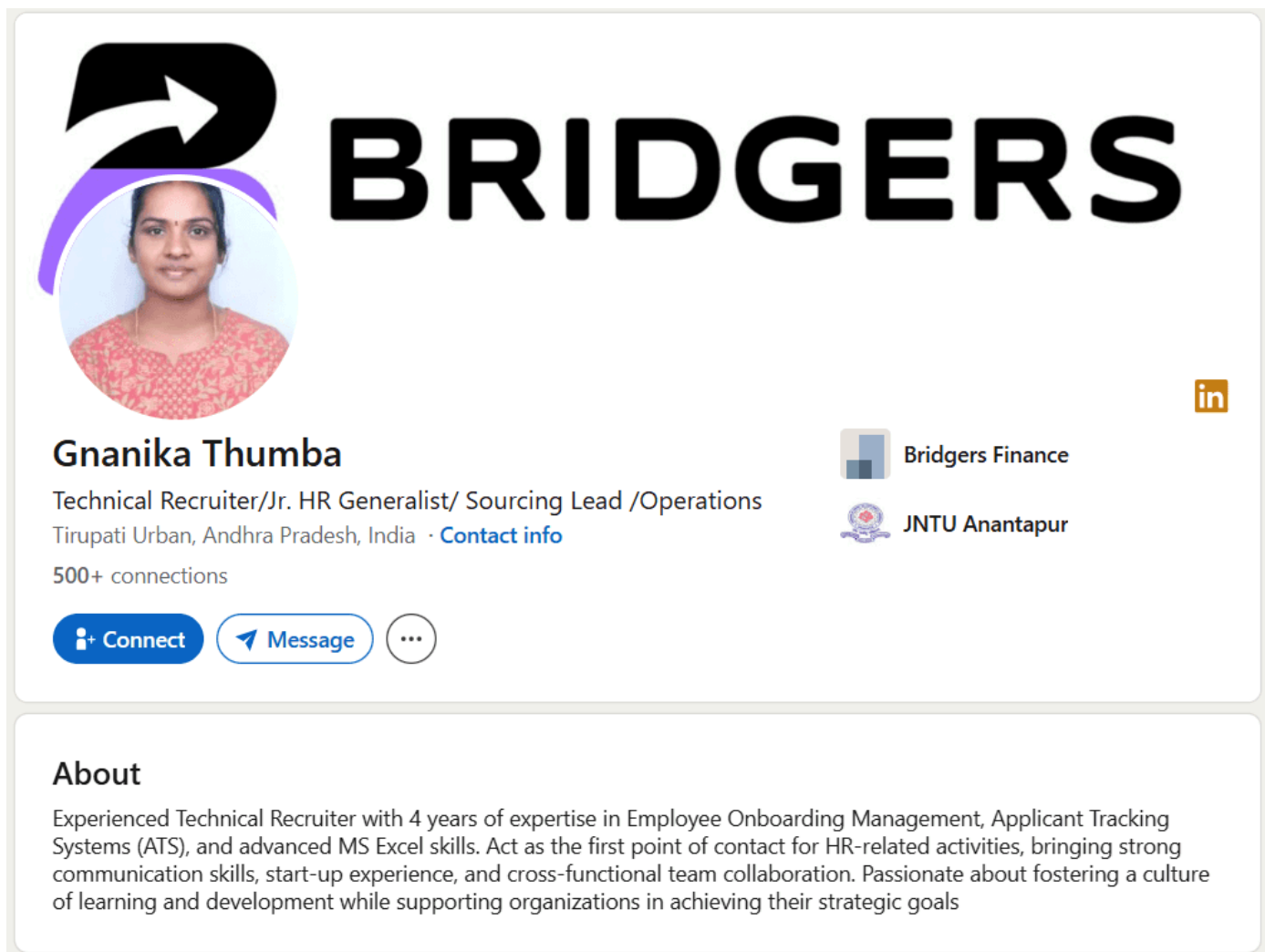


Figure 1. Fake LinkedIn profile used to publish fictitious job offerings

What is very interesting is that the launching of the Bridger Finance branch was slightly delayed because of the publishing of our initial blog post. Threat actors had the fake identity already prepared and several fake job offerings published and then a day or two after the blog was published the LinkedIn account was deactivated. This was part of the major cleanup of frontend and backend activities conducted by the threat actor operating the campaign. After a month passed, the LinkedIn profile was reactivated, and the campaign started again.

The same cleanup activity was observed at the same time with Blockmerce, the second fake crypto company that is being tracked as part of this campaign.



Blocmerce

Blocmerce is a blockchain technology company focused on transforming real-world commerce through secure, scalable, and m
b

1 follower
United States of America
<https://www.blocmerce.com>
[company/blocmerce](#)
info@blocmerce.com

Popular repositories

ccxt

Public

Forked from [ccxt/ccxt](#)

A JavaScript / TypeScript / Python / C# / PHP / Go cryptocurrency trading API with support for more than 100 bitcoin/altcoin exchanges

Python

test-orchestrator

Public

TypeScript

Repositories

Type ▾

Language ▾

Sort ▾

test-orchestrator

Public

TypeScript

0

0

0

0

Updated on Aug 12, 2025

ccxt

Public

Forked from [ccxt/ccxt](#)

A JavaScript / TypeScript / Python / C# / PHP / Go cryptocurrency trading API with support for more than 100 bitcoin/altcoin exchanges

Python

0

MIT

9,066

0

0

Updated on Jul 19, 2025



Figure 2. Fake Blocmerce GitHub organization

Like the Veltrix campaign, this branch was active in the second half of 2025. What is especially interesting in this part of the campaign is the effort that threat actors put into making Blocmerce appear to be legitimate. Those activities include common practices like the creation of a fake web site, social media profiles (X and LinkedIn) for the company. But the threat actors went a step further: establishing a legitimate legal entity — a limited liability corporation (LLC) — for Blocmerce, filing [registration papers](#) (PDF) with the Florida Secretary of State’s Office in late August that list the names and addresses of Blocmerce’s (fake) employees.



Alex Miller • 2nd
Chief Executive Officer at Blocmerce
3mo •

[+ Follow](#) ...

Building the Future of Loyalty with Blockchain | Blocmerce



Blocmerce

16 followers

3mo •

[+ Follow](#)

How are you today? Tell us what you want to hear from us.

If you want to stay up-to-date with us, follow our Page.



Figure 3. LinkedIn activity of Blocmerce CEO Alex Miller

Florida Limited Liability Company
BLOCMERCE LLC

Filing Information

Document Number	L25000392646
FEI/EIN Number	NONE
Date Filed	08/25/2025
Effective Date	08/25/2025
State	FL
Status	ACTIVE

Principal Address

2123 SANDRALA DRIVE
SARASOTA, FL 34231

Mailing Address

2123 SANDRALA DRIVE
SARASOTA, FL 34231

Registered Agent Name & Address

MILLER, ALEXANDRE, REV.
2123 SANDRALA DRIVE
SARASOTA, FL 34231

Authorized Person(s) Detail

Name & Address

Title CEO

ZII MILLER, MASTER A, REV.
2123 SANDRALA DRIVE
SARASOTA, FL 34231



Figure 4. Blocmerce LLC company registration information

The fake employees include three persons: CEO Alexandre Miller, whose LinkedIn profile was connected to Anushi Verma, a fake recruiter profile used in the Veltrix branch of the campaign, and Dmytro Buryma and Karina Lesova. These last two virtual accounts we will describe in more detail later in the post. (Miller's LinkedIn profile was eventually removed in an apparent effort to remove evidence.)

Article V

The name and address of person(s) authorized to manage LLC:

Title: CEO
MASTER A ZII MILLER REV.
2123 SANDRALA DRIVE
SARASOTA, FL. 34231 US

Title: MGR
DMYTRO BURYMA
7733 HOLIDAY DR
SARASOTA, FL. 34231 US

Title: MGR
KARINA LESOVA
7733 HOLIDAY DR
SARASOTA, FL. 34231 US

L25000392646
FILED 8:00 AM
August 25, 2025
Sec. Of State
fjeggleston

Article VI

The effective date for this Limited Liability Company shall be:

08/25/2025

Signature of member or an authorized representative

Electronic Signature: ALEXANDRE MILLER

I am the member or authorized representative submitting these Articles of Organization and affirm that the facts stated herein are true. I am aware that false information submitted in a document to the Department of State constitutes a third degree felony as provided for in s.817.155, F.S. I understand the requirement to file an annual report between January 1st and May 1st in the calendar year following formation of the LLC and every year thereafter to maintain "active" status.



Figure 5. Blocmerce LLC authorized representatives

Of course, these could be real people — even the operatives behind the campaign. But it is more likely that these are fake (or stolen) identities. That's a technique used by North Korean threat actors as described in a threat intelligence [research](#) from GitLab and other firms.

When we searched for information about residents who live at the addresses listed for these Bloomerice employees in the state filings, we found that the physical addresses are legitimate, but belong to different people.

What was the reasoning behind the attackers registering Blocmerce with Florida's Secretary of State? It remains unclear. Practically speaking, there is not much for the threat actor to gain from that. Trust from potential victims can easily be built with some LinkedIn DMs, but without official corporate filings.

The backend operations

The most significant shift between the first campaign we detected and these later campaigns was related to the backend operations of the campaign. Specifically: Threat actors started using another technique for hosting malicious dependencies. In the initial campaign, the fake interview tasks included a *package.json* file containing a dependency to a malicious package hosted in open source package managers like npm or PyPI. In the newest iterations, that changed. The malicious dependency is now hosted as a release artifact in GitHub repositories. At the same time, the OSS package managers were removed from the operation, at least in the analysed branches of the campaign.

This is likely made to minimize the detection risk. OSS package managers have been in the spotlight of the latest supply chain attacks and that has also attracted the attention of security vendors. Repositories like npm and PyPI are heavily monitored. From the attacker's perspective, that makes them an unnecessary risk. Their solution: move malicious dependencies to GitHub, which is not monitored nearly as much as the OSS repositories.

How was this migration performed? The reference to the malicious dependency was moved from *package.json* to the *package-lock.json* file. The list of dependencies in the lock files includes only the legitimate packages.

devops-challenge-sh / package-lock.json 

 craig-bridgers initial commit

Code Blame 867 lines (867 loc) · 29.6 KB

```
1  {
2    "name": "ss",
3    "version": "1.0.0",
4    "lockfileVersion": 3,
5    "requires": true,
6    "packages": {
7      "": {
8        "name": "ss",
9        "version": "1.0.0",
10       "license": "ISC",
11       "dependencies": {
12         "cors": "^2.8.6",
13         "dotenv": "^17.2.1",
14         "express": "^5.1.0"
15       }
16     },
```



Figure 6. Clean dependency list in package-lock.json file

The reference to the malicious dependency is buried deep inside the list of the transitive dependencies. The `resolved` field in the `package-lock.json` file instructs the package manager where to fetch specific package dependencies from. While all other dependencies are fetched from the official npm registry, the malicious one is fetched directly from a release artifact located in a crafted GitHub repository (Figure 7).

The attackers also created a GitHub repository with the same name as the repository belonging to a legitimate dependency and typo-squatted the maintainers' names to make the phishing attack harder to detect. In one of the GitHub repositories belonging to victims, for example, a reference to `ljharb/side-channel-weakmap` was observed. That's a typo-squatted package imitating [ljharb/side-channel-weakmap](https://github.com/ljharb/side-channel-weakmap), with a lowercase "L" replacing an uppercase "I" in the legitimate and popular GitHub package.

It would be difficult for a developer to spot that subtle difference between the lowercase 'l' and the capital 'I'. An interesting fact is that Jordan Harband, the maintainer behind the legitimate `ljharb` GitHub user account, has been directly [targeted in the recent phishing campaign](#) that resulted in compromise of the axios npm package maintainer's account. That campaign has also been attributed to North Korean state-sponsored threat actors, making him a popular target for that group.

```
779 },
780   "node_modules/side-channel-weakmap": {
781     "version": "1.0.2",
782     "resolved": "https://github.com/ljharb/side-channel-weakmap/releases/download/1.0.2/side-channel-weakmap-1.0.2.tgz",
783     "license": "MIT",
784     "dependencies": {
785       "call-bound": "^1.0.2",
786       "es-errors": "^1.3.0",
787       "get-intrinsic": "^1.2.5",
788       "object-inspect": "^1.13.3",
789       "side-channel-map": "^1.0.1"
790     },
791     "engines": {
792       "node": ">= 0.4"
793     },
794     "funding": {
795       "url": "https://github.com/sponsors/ljharb"
796     }
797   },
```



Figure 7. Malicious dependency resolved from a typosquatted GitHub release artifact.

Like in the original graphalgo campaign, the dependency was installed during the project setup and execution of the `npm install` command in one of the setup scripts.

To change the malicious payload for the latest campaigns, however, the threat actor only needs to push a new release artifact to the GitHub repository. That's a change that no one is likely to notice and it is much safer than pushing a new version of a package to npm that automatically gets picked up and analyzed by a dozen of security tools.

Several GitHub repositories containing malicious release artifacts were detected to be part of the campaign. Git log information for every one of them contains only one commit with GMT+9 timestamp (North Korea's time zone). The collected information indicates that this technique was used in the campaign from September 2025.

The content of the malicious dependencies is an npm package containing an encrypted, multi-stage downloader that eventually retrieves the final payload. That payload is the same RAT that we observed in the initial graphalgo campaign and described in [RL's first graphalgo blog post](#). The structure of the downloader code is pretty much the same as we observed in the earlier campaign, also.

One difference is the addition of new techniques to report the successful infection. Upon successful infection, a random wallet address is generated and a notification is sent to hardcoded and encrypted telegram and slack channels. The generated wallet address is also forwarded as an argument to the *addAddress* function defined in a smart contract identified by *0x7526aCdCF0B22f9B8F790CF069E5dD16CC414B0e* address on the sepolia testnet.

Sepolia Testnet
Search

Contract
0x7526aCdCF0B22f9B8F790CF069E5dD16CC414B0e

Source Code

Overview

ETH BALANCE

0 ETH

More Info

CONTRACT CREATOR

0x87BF60FB...Bb5a8FcF4 | 22 days ago

Transactions
Token Transfers (ERC-20)
Contract
Events

Code
Read Contract
Write Contract

Connect to Web3

Read Contract Information

1. getAddress (0xb93f9b0a)

2. getAddresses (0xa39fac12)

3. getAddressesLength (0xcbf78f47)

4. owner (0x8da5cb5b)

0x87BF60FB6657d5E5CD425E36FF18aa7Bb5a8FcF4 address



Figure 8. Sepolia testnet contract used in campaign

The contract was created by the `0x87BF60FB6657d5E5CD425E36FF18aa7Bb5a8FcF4` address. The transaction history for that address displays the evolution of backend infrastructure. The first [transaction](#) is inbound from `0x6EFB29CEe3b414272EB7A8F3Ebabf873D36bC033` and transfers 0.05 ETH to the address responsible for creation of the contract. These are likely the [free testnet tokens](#) that can be used to test smart contracts, meaning no additional cost for threat actors, and

no threat actor controlled address used for the initial funding. Then a test contract containing `addString` function is created and a few transactions testing its functionality are made before the contract used in the campaign is eventually created.

Everything is similar to the initial Veltrix branch of the campaign:

- The second stage downloader payload is then downloaded from a previously encrypted URL, `https://huvaret[.]art/public/startup.js?type=npm&ver=1.2`.
- That obfuscated javascript downloader fetches the last stage RAT from `https://huvaret[.]art/public/index.js?type=npm&ver=1.2`.

It is worth noting that there are also new npm packages that are being published as part of this campaign on a weekly basis. The packages have the same old “graph” themed names and use the same blockchain contract as described in this research. It remains unclear which frontend operation they belong to. It is possible that other Git-based platforms like GitLab or Bitbucket that have not been detected are being used in the campaign.

New technique: Git log rewriting

An interesting technique used to make the deception more convincing was observed in the GitHub repositories belonging to the GitHub organization that impersonates the SWFT Blockchain platform. There, the attacker created copies of legitimate repositories and rewrote their entire git history to make them look like they were conducted from attacker controlled accounts. An example of this can be seen in the `party-bridges-common` repository maintained by the fake `swft-blockchain` GitHub organization (Figure 9).

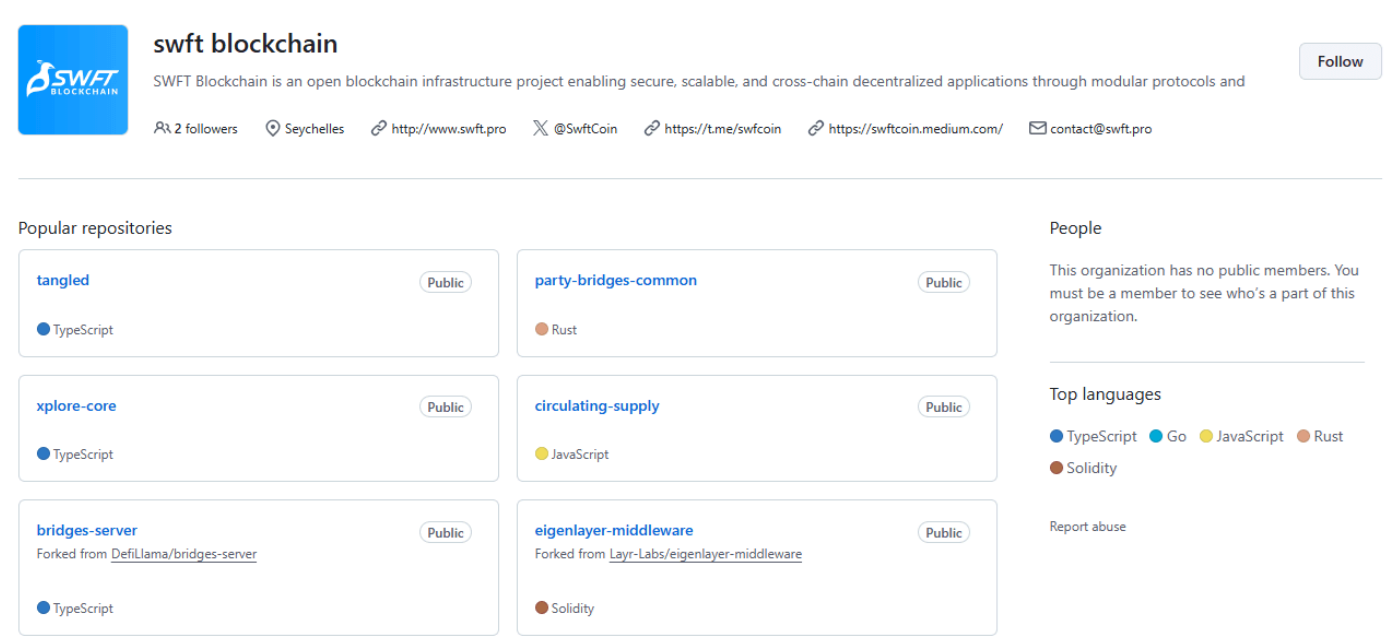


Figure 9. Repositories listed in threat actor controlled swft-blockchain GitHub organization

The *party-bridges-common* repository is a copy of a legitimate [parity-bridges-common](#) repository (*parity* → *party*) maintained by the [paritytech](#) organization. Comparison of repository activities and commits reveals the use of the git log rewriting technique.

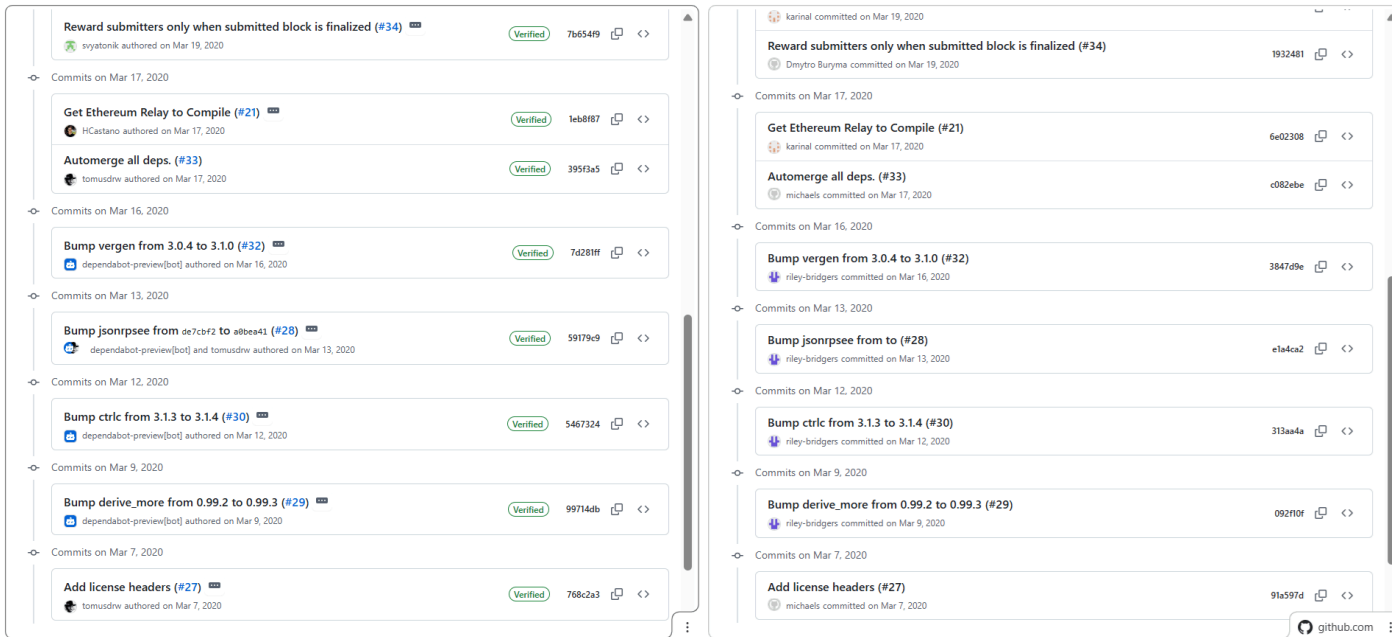


Figure 10. Commit info comparison between legit and faked repositories

Every commit is present in both repositories and has the same timestamp, but the commits from the typo-squatted repository don't have the verified badge and belong to threat actor's accounts. If you take a closer look, you will notice two names we mentioned earlier in the Blocmerce branch of the campaign: Dmytro Buryma and Karina Lesova. These Blocmerce employee accounts never existed in GitHub and are present only in local history of the fake repositories. This activity was likely conducted to build-up trustworthiness of threat actor controlled GitHub organizations.

But there are some GitHub accounts that were actually created and used to submit code in repositories belonging to the *swift-blockchain* GitHub organization. These include *riley-bridgers* and *craig-bridgers*. They suggest a connection to the "Bridgers Finance" branch of the frontend campaign.

Attribution details

The level of details that are taken care of as part of the campaign preparations is another proof point in the direction of a determined and organized state-sponsored threat actor.

And there's the use of fake or stolen identities and Git history rewriting of copied GitHub projects to increase credibility of such identities is also a new tactic in this campaign. The reuse of the same final payload and release artifacts as the initial graphalgo campaign, along with timestamps that align with North Korean time zones further support the prior attribution to the state-sponsored threat actors.

Developer ecosystem in the hot seat

Recent [supply chain attacks conducted by TeamPCP](#) have thoroughly shaken the already fragile trust in the open-source ecosystem and its poor security foundations.

The latest operations conducted by North Korean state-sponsored threat actors include the [compromise of one of the most popular npm packages: axios](#), as well as the theft of \$285M in the [Drift compromise](#). These incidents are a reminder that North Korean threat actors present the biggest threat to the entire OSS ecosystem.

The threat actors' use of advanced techniques and tactics, as well as an astonishing level of campaign preparation (setting up a Florida LLC) and their ability to adapt, makes North Korean threat actors a top threat to organizations or individual developers focused on cryptocurrency.

The RL research team's advice is to take special care when it comes to the third party software and development tools used in your organization. Recent weeks show that the security of open source dependencies and tools are a big security risk for most software development organizations. Practically, that means that the decades-old development model that assumes trust and equates popularity with security and integrity is broken.

Today, it is irrelevant if you are using a popular open source project that has hundreds of previous versions published without any security incidents. Its next version could easily contain a malware dropper or a wiper that takes out your entire organization. While that might shock you, campaigns like graphalgo mean that it is not a risk you didn't see coming.

Finally, if you are involved in the cloning or testing of a third-party GitHub project, approach it as if it was a malicious package, because there is a good chance that it actually is. Threat actors are finding novel techniques to hide inclusion of malicious code or dependencies in projects, and always come up with new auto-execution methods. To counter that threat, do your cloning and testing in a sandboxed or virtual environment and use dedicated security tools like [Spectra Assure](#) to verify software packages you are planning to use in your projects.

Join RL's free [Spectra Assure Community](#) to leverage advanced binary analysis to discover the newest open-source threats and malicious packages like the ones from this campaign.

Indicators of Compromise (IOCs)

Indicators of Compromise (IOCs) refer to forensic artifacts or evidence related to a security breach or unauthorized activity on a computer network or system. IOCs play a crucial role in cybersecurity investigations and cyber incident-response efforts, helping analysts and cybersecurity professionals identify and detect potential security incidents.

The following IOCs were collected as part of RL research team investigation of the graphalgo software supply chain campaign.

Network indicators

huvaret[.]art

Blockchain addresses

0x7526acdcf0b22f9b8f790cf069e5dd16cc414b0e

0x87bf60fb6657d5e5cd425e36ff18aa7bb5a8fcf4

Downloaded stages

File type	SHA1
JavaScript	5c30d58dc44182f959c8035e990153b3553deace
JavaScript	f1487451933a05a680e71dde7a2b11560d2d33a7

Resolved GitHub release artifacts

URL

<https://github.com/expressjs/body-parser/releases/download/v2.2.0/body-parser-2.2.0.tgz>

<https://github.com/expressjs/body-parser/releases/download/v2.2.1/body-parser-2.2.1.tgz>

<https://github.com/expressjs/body-parser/releases/download/v2.2.1/body-parser-2.2.1.tgz>

<https://github.com/ljharb-js/parseurl/releases/download/v1.3.3/parseurl-1.3.3.tgz>

<https://github.com/ljharb/side-channel-weakmap/releases/download/1.0.2/side-channel-weakmap-1.0.2.tgz>

Package indicators:

package_name	version	sha1	repository
--------------	---------	------	------------

graph-dynamic	1.0.0	eea702ebc53a4b9f8c1b511fffce16f6874de666	npm
graph-dynamic	1.0.1	e3a71d70a5a5d3790a352955edb3bb7a003dd6d5	npm
graphbase-js	2.1.4	e4bf38b28b7aeec2685d1d2581d271c965ee6b84	npm
graphbase-js	2.1.5	f6c574baf05234284966abba25377eee589bba6a	npm
graphbase-js	2.1.6	d75b3abdd7af3b18be945caa721f1e4e076146c	npm
graphbase-js	2.2.1	679fdccecfed0e5cc2c2636fe649a668d50f63ea	npm
graphcore-js	2.3.2	7a35c8b0e1182b1fd12a8acb49cfeab22eae1d6	npm
graphcore-js	2.3.4	7af1065e7e6fb6184f99541d142132ba6db03a41	npm
graphcore-js	2.4.1	65de94d3eb0524fc17df5fdec8c20afada2d0119	npm
graphcore-js	2.4.2	173bb313e6e29525fd6b04407c1c6e8a4a29c7a0	npm
graphlib-js	1.2.1	c4326153401904e82b17726864be65cac0c97fd1	npm
graphlib-js	1.2.2	d531769223f468f93e42e19dea74cb16443ba0b8	npm
graphlib-js	1.3.2	cb7ac56cf1c3c1aac9fe4c86a9a323be0698de6c	npm
graphlib-js	1.3.3	ebb4630024764bdf5e5c1013166cc461d3df7550	npm
graphlib-js	1.3.4	c7692a6816cc0eb61216358ff0367d7469125192	npm

Tags: [Threat Research](#)