

IRSTI 50.37.23

N. Abdinurova<sup>1</sup>, M. Sokitbayev<sup>2</sup>, T. Suyerbassov<sup>3</sup>, A. Baimolda<sup>4</sup>  
<sup>1,2,3,4</sup>Suleyman Demirel University, Kaskelen, Kazakhstan

## VULNERABILITIES IN CI-PLATFORMS THAT CAN BE EXPLOITED FOR CRYPTOCURRENCY MINING AND POSSIBLE SOLUTIONS TO THIS PROBLEM

**Abstract.** The topic of how vulnerabilities on CI-platforms that are subject to cryptocurrency mining is relevant. One solution is that the github has manually set the commit confirmation so that the cryptominer cannot automate mining.

**Keywords:** component, formatting, style, styling, insert.

\*\*\*

**Аңдатпа.** Криптовалютаны өндіруге жататын CI-платформаларындағы осалдықтар тақырыбы өзекті болып табылады. Шешімдердің бірі - GitHub келісімді растауды қолмен орнатты, осылайша криптовалюта өндірушісі майнинг өндіруді автоматтандырмайды.

**Түйін сөздер:** компонент, пішімдеу, стиль, стильдеу, кірістіру.

\*\*\*

**Аннотация.** Тема о том, насколько актуальны уязвимости на CI-платформах, которые подвержены добыче криптовалют, актуальна. Одно из решений заключается в том, что GitHub вручную установил подтверждение фиксации, чтобы криптомайнер не мог автоматизировать майнинг.

**Ключевые слова:** компонент, форматирование, стиль, оформление, вставка.

### *Introduction*

At the moment, developers are suffering from the actions of their unscrupulous colleagues. And this is not due to the direct effect of some on others. Some developers have started secretly mining cryptocurrency on various free CI platforms, using their computing power. Why this happens is pretty clear-making money. There are cases when such miners earned a monthly salary for a month, while all they needed was Internet access and a computer. Of course, such actions could not fail to attract attention from the administration of these sites. And accordingly, the fight against miners has also affected ordinary users of CI platforms. So why is it not so easy to detect unwanted actions, and what methods already exist to combat mining? In order to answer

these questions, you need to first understand how CI platforms and mining systems work.

## *II. Background materials*

Most cryptocurrencies utilize proof-of-work for mining. PoW-tasks are not initially planning for a individual, their arrangement by a computer is continuously achievable in a limited time, but it requires a parcel of computing control. At the same time, checking the coming about arrangement requires much less assets and time. This definition will be valuable to us to discover a arrangement to the issue with diggers.

Identify applicable funding agency here.

If none, delete this. Moreover, in expansion to the classic mining strategies, there are more exceptional ones. For case, mining on phones, SSD drives, and indeed within the computer browser. The goal of CI is to supply a steady and robotized way to construct, bundle, and test applications. CI stages offer assistance engineers make completely utilitarian websites by making see environments and automatically running E2E tests for them. This can be called CI (Nonstop Integration). VMs on CI-platform utilize memory snapshotting to work like Docker holders. They're quicker and more developer friendly than a conventional VM. Benefits: simple to form organizing situations as required, and they can be spun up in seconds since the modern VM contains a duplicate of the database that was set up within the final run. E2E tests can be run in parallel rapidly and effectively on the off chance that you'll copy the whole test VM, sparing important time for developers We can make handfuls of VMs per department, since we consequently sleep them when they're not in utilize.

## *III. Literature review*

In this review they perform a comprehensive examination on Alexa's Best 1 Million websites to shed light on the prevalence and efficiency of assault that mining the cryptocurrency. They think almost the websites impacted by drive-by mining to induce it the methods being utilized to maintain a strategic distance from area, and the foremost later web propels being mishandled to beneficially mine cryptocurrency. They depict how to detect cryptocurrency mining on the location and their approach may well be coordinates into browsers to caution clients around quiet cryptomining when going to websites that don't inquire for their assent.

## *IV. Methods and materials*

Since developers can run arbitrary code on these servers, they often violate the terms of service in order to run cryptocurrency miners as a "build stage" for their websites. For example, the repository "testronan/MyFirstRepositoryFlask". The user "testronan" makes a commit every hour. The software developer uses 5 platforms at once to test their routing.

These are TravisCI, CircleCI, GitHub Actions, Wercker, and LayerCI. His SI tasks run “listen.sh”: a shell script that reduces a difficult NodeJS script with, at first glance, random numbers. However, he is quite experienced in writing scripts:

```
(sleep 10; echo 4; sleep 2; echo "tex.webd";sleep 2; echo 7; sleep 1; echo 1; sleep 1; echo "exit"; sleep 2) | stdbuf -oL npm run commands
```

“MyFirstRepository-Flask” does not contain anything shared with Flask or web servers. It contains scripts for mining cryptocurrency that send WebDollars to an unnamed online wallet. And these numbers are needed for setting up options, for implementing WebDollar on NodeJS.

The repository does not use github resources, for example. Instead, it abuses the “cron” function in order to make one commit every hour and mine webdollar on these platforms. The user “vippro99” does not hide his intentions at all. Of the 10 repositories, the majority is related to either cryptocurrency or browser automation.











The nodejs-monney repository has all sorts of scripts for running Chrome instances with the well-known Google puppeteer project. This puppeteer project allows you to run the browser through commands, as well as test your applications on them.







The idea is simple: cryptocurrency mining in CI is quite simple to detect (for example, with support for analyzing executable content), but browser automation is an ordinary workload in CI.

```
puppeteer.launch({ headless: true, args: ['--no-sandbox',  
  '--disable-setuid-sandbox', '--window-size=500,500', '--user-agent=Mozilla/5.0  
(Macintosh; Intel Mac OS X 11_2_3) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/89.0.4389.90 Safari/537.36' ] })  
.then(async browser => {  
  console.log('-- Running chrome!!');  
  const page = await browser.newPage();  
  await page.goto('https://vippro99.github.io/-meocoder-nodejs-tool');  
  page.on('console', (msg) => console.log(msg.text()));  
  await page.waitForTimeout(((Math.floor(Math.random()*6)+52)*60)*1000);  
  await browser.close();  
})
```

This GitHub Pages site has a simple browser-based miner of the Monero cryptocurrency, which resembles Coinhive in its idea.

## V. Data and result

Status	Pipeline	Triggerer	Commit	Stages	Duration
	#304401483 latest error		master -> e6f8e6f6 update		2 minutes ago
	#304401356 error		master -> dc58c457 update		3 minutes ago
	#304401172 error		master -> b6d796cf update		4 minutes ago
	#304399201 error		master -> 232bf081 added Test.txt		13 minutes ago
	#304398095 error		master -> 97b7147d added .gitlab-ci.yml f...		17 minutes ago

Name
 .idea
 out/production/OpenWeather/c...
 src/com/company
 .gitlab-ci.yml
 OpenWeather.iml
 Test.txt

Continuous integration works by sending small chunks of code to your program's code base in the Git repository, and each time it clicks, it launches a pipeline pipeline to compile, test, and verify code changes before merging them into the main branch. Continuous delivery and deployment consist of the next step of CI, deploying your application in a production environment each time you click on the default repository branch. These methodologies allow you to detect errors in the early stages of the development cycle, ensuring that all code deployed in the production environment meets the code standards set for your application.

Securing your applications and infrastructure is a major challenge for most organizations. But you need to balance the momentum of continuous development, maintenance, and deployment with the challenges of keeping these tasks secure and keeping your deployed programs and infrastructure secure. As an optimized practice, DevOps automated continuous integration/continuous delivery (CI/CD) pipelines using tools such as Jenkins, Concourse, or CircleCI allow you to create assembly, testing, and deployment systems with extensive customization capabilities. And you're not limited to compiling and testing: most CI/CD tools allow you to connect the functionality of other tools, such as performance profilers and error monitoring. Taking a step forward in DevOps in Infrastructure as Code (IaC) operations and practices, many teams now use CI/CD pipelines to manage configuration files to build the infrastructure on which their applications run. This means that CI/CD tools also handle change

management, testing and deployment of Docker files, Kubernetes manifests, Helm charts, etc. While most CI/CD discussions focus on program code - preventing errors and crashes - moving to IaC configuration management with CI/CD tools highlights another potential challenge: managing the secrets needed to run your applications and systems, the secrets that may end up endangering and putting your systems at risk. Automating build, testing, and deployment tasks removes much of the need to store and track secrets from your developers and administrators, which is a valuable first step, but shifts responsibility for those secrets to your CI/CD pipeline and related tools. In this article, we will look at some general concepts and best practices for ensuring the efficient and secure use of authentication and authorization secrets in CI/CD pipelines. The first steps in securing your team's CI/CD pipeline include blocking configuration managers, repository-based systems, and assembly servers. The pipeline should be monitored from start to finish, ensuring waterproof access control throughout the tool chain. Script builds should be checked for vulnerabilities, and source code should be regularly checked for vulnerabilities before deploying the application in a production environment. Security of secrets must be applied both during transportation and at rest. Best practices include the following: Remove hard-coded secrets from Jenkinsfiles and related CI/CD configuration files. Have strict security settings, such as one-time passwords, for secrets related to more important tools and systems.

Distribute secrets among Jenkinsfiles to reduce the potential target of the attack for each file. Use password managers and change passwords after each use. Know who has access to what. Whether based on roles, time, or tasks, there must be a clear access control repository. Another option to consider is to segment secrets based on access levels. Machine identification is crucial to protect access to non-human containers. Typically, the authenticator certifies that the attributes of the container runtime of the container requester client match the intrinsic characteristics of the valid container. After authentication, the container can access multiple resources based on defined rolebased access control policies. You must destroy containers and virtual machines after use. Make sure that secrets are not inadvertently passed during the build process for extraction requests through your CI/CD pipelines. Expand the practice of minimum privileges: provide access only to the necessary secrets. In addition to employee access, the least privileges also apply to applications, systems, or connected devices that require privileges.

#### *VI. Conclusion*

In addition to quickly finding mining in the code, there are other ways. One of the solutions that GitHub has taken is to manually confirm commit. That is not difficult for the developer, since commits are not made so often, unlike repositories that are aimed at mining. This slows down the development process a bit, but brings significant hassle to them, unlike the next option. Companies that provide free CI for individual programmers can become paid, or simply

limited in operation. CI platforms such as LayerCI, GitLab, TravisCI, and shippable degrade or even shut down their free servers due to hidden mining attacks. There is also a longerterm option. This is a transition from proof-of-work to proof-of-stake. A method of protection in cryptocurrencies, in which the probability of a participant forming another block in the blockchain is proportional to the share that belongs to this participant. This means that now there will be no need to spend the computing power of the systems, thus this option will repel miners from CI systems. PoS is used in such cryptocurrencies: Ethereum, Nxt and BlackCoin.

### **References**

- 1 On the Security and Performance of Proof of Work Blockchains Arthur Gervais, Ghassan O. Karame, Karl Wust, Vasileios Glykantzis, Hubert Ritzdorf, Srdjan Capkun. Available: <https://eprint.iacr.org/2016/555.pdf>.
- 2 Securing Proof-of-Stake Blockchain Protocols Wenting LiSebastien, AndreinaJens-Matthias, Bohli Ghassan Karame. Available: <https://www.springerprofessional.de/en/securingproof-of-stake-blockchain-protocols/15047868>.
- 3 Continuous Integration and Its Tools, Mathias Meyer. Available: <https://ieeexplore.ieee.org/document/6802994>.
- 4 Mine Sweeper: An In-depth Look into Drive-by Cryptocurrency Mining and Its Defense. Available: <https://dl.acm.org/doi/10.1145/3243734.3243858>.