10.11.2025).
2. National Institute of Standards and Technology. Generative AI Profile for the NIST AI RMF (NIST AI 600-1). 2024. URL: https://nvlpubs.nist.gov/nistpubs/ai/NIST.AI.600-1.pdf (дата звернення: 10.11.2025).
3. NSA; CISA; FBI. Contextualizing Deepfake Threats to Organizations. 2023. URL: https://media.defense.gov/2023/Sep/12/2003298925/-1/-1/0/CSI-deepfakethreats.pdf (дата звернення: 10.11.2025).
4. European External Action Service. 2nd Report on Foreign Information Manipulation and Interference (FIMI). 2024. URL: https://www.eeas.europa.eu/sites/default/files/documents/2024/EEAS-2nd-Report%20on%20FIMI%20Threats-January-2024_0.pdf (дата звернення: 10.11.2025).
5. Osadchuk R. AI tools usage for disinformation in the war in Ukraine. Digital Forensic Research Lab. 09.07.2024. URL: https://dfrlab.org/2024/07/09/ai-tools-usage-for-disinformation-in-the-war-in-ukraine/ (дата звернення: 10.11.2025).

Захист CI/CD-конвеєрів автоматизованого оновлення Docker-контейнерів на основі криптографічного підписування образів

УДК 004.056:004.75

Віталій Тимощук[1], Дмитро Тимощук[2]

Тернопільський національний технічний університет імені Івана Пулюя,
[1]tymoshchuk@tntu.edu.ua, [2]dmytro.tymoshchuk@gmail.com

Розробка програмного забезпечення дедалі частіше базується на використанні контейнеризації та автоматизованих CI/CD-конвеєрів. Docker-контейнери дають змогу швидко збирати, доставляти й оновлювати застосунки, однак водночас створюють нові ризики для безпеки. Особливо критичною є проблема довіри до контейнерного образу, який автоматично завантажується з реєстру та запускається у продуктивному середовищі. У типовій схемі CI/CD новий образ збирається після зміни коду, публікується в container registry, після чого сервер отримує його за тегом, наприклад latest. Такий підхід є зручним, але не гарантує, що завантажений образ справді відповідає легітимній версії застосунку.

Основна загроза полягає в тому, що контейнерний образ може бути підмінений у реєстрі, у процесі доставки або через помилкову конфігурацію CI/CD-конвеєра. У такому випадку продуктивне середовище може запустити шкідливий або застарілий образ як легітимне оновлення. Це безпосередньо пов’язано з атаками на ланцюг постачання програмного забезпечення, кількість яких суттєво зростає внаслідок активного використання автоматизованих засобів збірки, сторонніх залежностей і container registry [1]. Традиційні засоби мережевого захисту не завжди здатні виявити такі атаки, оскільки скомпрометований образ може виглядати як звичайне оновлення.

Метою роботи є підвищення безпеки автоматизованого оновлення Docker-контейнерів шляхом використання криптографічного підписування

контейнерних образів, перевірки їх автентичності перед розгортанням і прив'язки образів до незмінного SHA-256 дайджесту. Запропонований підхід орієнтований на інтеграцію в наявний CI/CD-процес без суттєвої зміни його логіки.

У межах запропонованої моделі контейнерний образ після збірки не лише публікується в реєстрі, а й підписується за допомогою інструмента Cosign. Для цього використовується keyless-підхід Sigstore, що дає змогу виконувати підписування без зберігання довготривалих приватних ключів. У середовищі GitHub Actions автентифікація може здійснюватися через OIDC, після чого сервіс Fulcio видає короткоживучий сертифікат для підписування конкретного артефакту [2]. Це знижує ризики компрометації ключів і спрощує впровадження механізмів підпису в CI/CD.

Важливою складовою підходу є використання адресації образів за криптографічним дайджестом, а не лише за тегом. Теги на кшталт latest є змінними й можуть у різні моменти часу вказувати на різні образи. Натомість SHA-256 дайджест однозначно ідентифікує вміст контейнерного образу. Якщо образ змінено хоча б мінімально, його дайджест також зміниться. Тому використання формату repository@sha256:<digest> забезпечує отримання сервером саме того артефакту, який відповідає зафіксованому дайджесту, а в поєднанні з перевіркою цифрового підпису дає змогу підтвердити, що цей артефакт був зібраний і підписаний у CI/CD-конвеєрі [3].

Запропонований процес складається з двох основних етапів. На першому етапі CI/CD-конвеєр виконує збірку Docker-образу, публікує його в container registry, отримує SHA-256 дайджест і підписує образ за допомогою Cosign. До підпису можуть додаватися анотації, які фіксують контекст збірки: ідентифікатор workflow, номер запуску, commit hash і час створення образу. Це підвищує прозорість і дає змогу встановити, з якої версії вихідного коду та в межах якого CI/CD-процесу був сформований артефакт.

На другому етапі, перед розгортанням, сервер виконує обов'язкову верифікацію підпису. Перевіряється не лише факт наявності підпису, а й відповідність підписувача очікуваному репозиторію та workflow. Якщо образ не має валідного підпису або був підписаний неавторизованим джерелом, процес розгортання зупиняється. Таким чином, навіть у разі компрометації контейнерного реєстру зловмисник не зможе непомітно впровадити власний образ, оскільки він не пройде криптографічну перевірку.

Окрему увагу приділено rollback-атакам. Така атака полягає у спробі повернути систему до старішої, але раніше підписаної версії образу, яка може містити вже відомі вразливості. Сам по собі цифровий підпис підтверджує автентичність артефакту, але не підтверджує його актуальність. Тому в запропонованій моделі передбачено політику контролю версій і часу підписання. Відкат дозволяється лише до попередньо відомих і перевірених стабільних версій, а не до довільного старого образу. Такий підхід дає змогу зберегти можливість легітимного rollback у разі помилки релізу та зменшує ризик використання rollback як каналу атаки.

Для перевірки ефективності підходу було змодельовано кілька типових сценаріїв: підміну образу в container registry, зміну тегу latest, спробу запуску

непідписаного образу, несанкціоноване додавання стороннього образу до довіреного реєстру та спробу зловмисного rollback. У базовому підході, де використовується лише тег без перевірки підпису, підміна образу може призвести до запуску шкідливого контейнера. У захищеній моделі розгортання прив'язується до конкретного SHA-256 дайджесту та перед запуском обов'язково виконується перевірка підпису. У результаті підроблені, непідписані або підписані неавторизованим суб'єктом образи блокуються ще до запуску.

Практична перевага запропонованого рішення полягає в тому, що воно не потребує радикальної перебудови CI/CD-процесу. Підписування й перевірка додаються як окремі автоматизовані кроки. При цьому накладні витрати на виконання криптографічних операцій є незначними порівняно із загальним часом збірки та розгортання контейнерного застосунку. Отже, модель може бути використана в DevSecOps-практиках як додатковий рівень захисту програмних артефактів [4].

Водночас криптографічне підписування не усуває всі можливі ризики. Якщо шкідливий код потрапив до образу ще до моменту підписання, наприклад через компрометацію CI-середовища, залежностей або runner-а, такий образ може бути формально валідно підписаний. Тому запропонований підхід доцільно поєднувати з іншими засобами безпеки: скануванням образів на вразливості, мінімізацією прав доступу в CI/CD, перевіркою сторонніх залежностей, code review та політиками контролю запуску контейнерів у середовищі виконання [5].

Отже, використання Cosign/Sigstore, адресації за SHA-256 digest і обов'язкової верифікації підпису перед розгортанням суттєво підвищує рівень довіри до автоматизованих оновлень Docker-контейнерів. Запропонований підхід забезпечує перевірку автентичності та цілісності контейнерних артефактів, а також створює основу для контрольованого керування їх версіями за рахунок використання незмінних digest і політик дозволених релізів. Це зменшує ризики атак на ланцюг постачання програмного забезпечення та може бути інтегроване в сучасні CI/CD-конвеєри без суттєвої зміни їх логіки.

1. Sonatype. *State of the Software Supply Chain Report: 10 Year Look*. URL: https://www.sonatype.com/state-of-the-software-supply-chain/2024/10-year-look.
2. Newman Z., Meyers J. S., Torres-Arias S. *Sigstore: Software Signing for Everybody*. CCS '22: ACM SIGSAC Conference on Computer and Communications Security, 2022. DOI: 10.1145/3548606.3560596.
3. RFC 6234. *US Secure Hash Algorithms: SHA and SHA-based HMAC and HKDF*. URL: https://datatracker.ietf.org/doc/html/rfc6234.
4. GitHub. *Sigstore Cosign: Code Signing and Transparency for Containers and Binaries*. URL: https://github.com/sigstore/cosign.
5. Koishybayev I., Nahapetyan A., Zachariah R., Muralee S., Reaves B., Kapravelos A., Machiry A. *Characterizing the Security of GitHub CI Workflows*. 31st USENIX Security Symposium, 2022. P. 2747–2763.