



DOI 10.28925/2663-4023.2026.35.1447

УДК 004.75; 004.056

Марценюк Євгеній

аспірант кафедри захисту інформації

Національний університет «Львівська політехніка», Львів, Україна

ORCID: 0009-0009-2289-0968

yevhenii.v.martseniuk@lpnu.ua

Маньковський Богдан

студент кафедри захисту інформації

Національний університет «Львівська політехніка», Львів, Україна

ORCID: 0009-0001-3857-9842

bohdan.mankovskyi.kb.2023@lpnu.ua

Дейнека Олег

аспірант кафедри захисту інформації

Національний університет «Львівська політехніка», Львів, Україна

ORCID: 0009-0005-9156-3339

deinekaoleh.86@gmail.com

**АВТОМАТИЗОВАНА СИСТЕМА КЕРУВАННЯ ЖИТТЄВИМ ЦИКЛОМ ЕТАЛОННИХ ОБРАЗІВ
ЯК ЗАСІБ ЗНИЖЕННЯ ВРАЗЛИВОСТЕЙ ПУБЛІЧНОЇ ХМАРНОЇ ІНФРАСТРУКТУРИ**

Анотація. Значна частина досліджень і практичних заходів із безпеки публічних хмар зосереджена на рівні окремого екземпляра — віртуальної машини, EC2-інстанса чи контейнера, — тоді як організаційні процеси керування машинними образами (image management) залишаються поза увагою. На практиці ці процеси або взагалі не впроваджені, або ігноруються, або сформовані підрозділом інформаційної безпеки некоректно, унаслідок чого до промислового середовища потрапляють застарілі, невідповідні та вразливі образи. У цьому дослідженні керування життєвим циклом еталонних образів (golden images) розглядається як окремий рівень контролю безпеки для монолітних застосунків, розгорнутих на віртуальних машинах у VMware vSphere, Google Compute Engine, AWS та Microsoft Azure. Наукова новизна роботи полягає в інтеграції конвеєра формування незмінних (immutable) образів на базі HashiCorp Packer та Ansible з оркестрацією Jenkins/Tekton, автоматизованим скануванням, шлюзом політик, реєстром доказів перевірки, моніторингом і погодженням через ITSM. Конкретні допоміжні платформи розглядаються як взаємозамінні реалізації логічних функцій, а не як самостійні об'єкти дослідження. Такий підхід переносить контроль вразливостей ліворуч (shift-left) — на етап складання образу — та закриває розрив підзвітності між командами DevSecOps, InfoSec і власниками застосунків. У роботі представлено авторську референсну архітектуру для монолітних VM-навантажень та результати контрольованого локального експерименту з чотирма історичними Ubuntu 24.04 cloud images. Парне порівняння трьох станів — baseline, full-upgrade і full-upgrade з контрольованим видаленням застарілих пакетів ядра — виконано за однакового APT snapshot і замороженої бази Trivy. Звичайний full-upgrade зменшив 1680 виправних HIGH/CRITICAL CVE до 1605, тобто забезпечив 4,46% покриття. Після перезавантаження в нове ядро та точного видалення 22 застарілих kernel packages кількість виправних HIGH/CRITICAL зменшилася до нуля без втрати перевіреної функціональності. Policy gate змінив рішення з BLOCK на REVIEW, оскільки в кожному образі залишилися п'ять CRITICAL CVE без доступного виправлення.

Ключові слова: публічна хмара; еталонний образ (golden image); машинний образ; монолітна архітектура; незмінна інфраструктура (immutable infrastructure); DevSecOps; HashiCorp Packer; сканування вразливостей; AWS AMI; Azure Compute Gallery; GCE Custom Images; VMware vSphere; керування життєвим циклом образів; shift-left.



ВСТУП

Публічні хмарні платформи стали основою сучасних корпоративних ІТ-інфраструктур, а масштаб їхнього використання зробив швидке та відтворюване постачання обчислювальних ресурсів критичною вимогою. Водночас переважна більшість наукових праць і практичних методик безпеки хмари зосереджена на рівні окремого екземпляра: конфігурації віртуальної машини, керуванні патчами операційної системи, захисті контейнерів під час виконання. Значно менше уваги приділяється джерелу, з якого ці екземпляри народжуються, — машинному образу (VM image, AMI, custom image, template). Саме образ визначає початковий стан безпеки сотень чи тисяч похідних екземплярів, і якщо він застарілий, непропатчений або сформований без контролю, кожен розгорнутий із нього ресурс успадковує його вразливості.

Проблематика дослідження. Організаційні процеси керування образами — їхнє формування, затвердження, версіонування, сканування та виведення з експлуатації — на практиці часто або не впроваджені, або впроваджені формально й ігноруються, або сформовані без чіткого власника, шлюзу схвалення та інтеграції сканування у конвеєр складання. Наслідком є неконтрольоване поширення «сніжинкових» (snowflake) образів, розбіжність (drift) між середовищами та накопичення застарілих артефактів. Галузеві звіти 2025–2026 років фіксують високу частку занедбаних хмарних активів і подальше зростання ролі експлуатації програмних вразливостей як початкового вектора атак.

Актуальність. Станом на 2026 рік актуальність проблеми підтверджують останні доступні галузеві дані: за Verizon DBIR 2026, 31% порушень безпеки починалися з експлуатації програмних вразливостей, яка випередила викрадені облікові дані як провідний спосіб початкового проникнення [26]. Водночас звіти за 2025 рік фіксують значну частку занедбаних і непропатчених хмарних активів, а атака whoAMI показала ризик підміни джерела AMI. Отже, актуальність у 2026 році обґрунтовується не «результатами за майбутній період», а найновішими опублікованими звітами та збереженням проблеми керування джерелами образів.

Мета роботи. Основною метою дослідження є розробка та обґрунтування архітектурного рішення — автоматизованого конвеєра формування, валідації та промоції еталонних образів для VMware vSphere, GCE Custom Images, AWS AMI та Azure Compute Gallery, — а також формування ризик-моделі й методики майбутньої кількісної оцінки його впливу на монолітні VM-навантаження.

Об'єктом дослідження є процеси постачання та керування життєвим циклом машинних образів для монолітних застосунків у публічній мультихмарній інфраструктурі організації. Предметом дослідження є методи та засоби автоматизованого формування незмінних еталонних образів, вбудованого сканування вразливостей і контрольованої промоції, що знижують рівень успадкованих вразливостей у похідних екземплярах.

АНАЛІЗ РИЗИКІВ І НАЯВНИХ РИНКОВИХ РІШЕНЬ

Сфера застосування та межі рішення

Запропонований підхід призначений для організацій, які централізовано постачають базові VM-образи в AWS, Microsoft Azure, Google Cloud або VMware vSphere, мають платформну/DevSecOps-команду, формалізовані вимоги InfoSec та можливість інтеграції з CI/CD, CMDB і ITSM. Цільовим типом навантаження є монолітний застосунок, у якому операційна система, runtime, бібліотеки, агенти безпеки та прикладний компонент постачаються й оновлюються як одна VM-одиниця.

У межі рішення входять відтворюване складання й гартування базових образів, сканування CVE та секретів, формування SBOM/provenance, контрольована промоція, допуск лише схвалених версій, депрекація та ініціювання заміни похідних VM. Поза первинними межами залишаються мікросервісні системи з контейнерною одиницею постачання, Kubernetes admission, повноцінний runtime EDR/CDR, безпека прикладного коду, мережевий захист і serverless packaging. Конвеєр не усуває zero-day і не оновлює вже запущені VM без повторного розгортання або заміни.

Проблема вразливостей на рівні машинних образів

Машинний образ є артефактом, що фіксує стан операційної системи, встановлене програмне забезпечення та конфігурацію у момент складання. Кожна з чотирьох розглянутих платформ надає власний механізм галереї образів — VMware vSphere templates, GCE Custom Images, AWS AMI та Azure Compute Gallery, — проте відповідальність за гартування, сканування, валідацію та керований життєвий цикл цих образів покладається на споживача. За відсутності керованого процесу образ швидко «стареє»: у ньому накопичуються відомі вразливості пакетів, а розгорнуті з нього екземпляри розходяться з джерелом (image drift). Нативні механізми публікації та обмеження використання образів зв'язано з документацією AWS, Azure, Google Cloud і VMware [19–22].



Фокус на монолітній архітектурі обрано тому, що одиниця постачання монолітного застосунку безпосередньо збігається з об'єктом дослідження — машинним образом. Вразливість у базовій ОС, runtime або системній бібліотеці успадковується всім застосунком і кожною VM, створеною з цього образу. Еталонний образ у такому середовищі дає єдину контрольовану базову лінію, спрощує відтворення, аудит і заміну всієї програмно-системної одиниці.

Це не означає, що монолітна архітектура універсально краща за мікросервісну. Для мікросервісів основним артефактом є контейнерний образ, а контроль здійснюється також на рівні реєстру контейнерів, оркестратора й admission policy. Включення цих механізмів змішало б два різні життєві цикли. У межах поставленої задачі моноліт є доцільнішим об'єктом перевірки завдяки однозначній відповідності «один еталонний VM-образ — одна версія застосунку», меншій кількості незалежних артефактів і простішій атрибуції виявленої вразливості до базового образу.

Звіт Orca Security 2025 використано для оцінювання накопичення застарілих активів: він повідомляє в середньому 115 вразливостей на хмарний актив, 32% занедбаних активів, а також поширеність давніх CVE та непропатчених вебсервісів. Ці показники характеризують масштаб проблеми керування життєвим циклом, але не є результатом нашого експерименту [1]. Звіт Wiz 2025 використано окремо для контексту експозиції даних: 54% досліджених хмарних середовищ містили доступні ззовні VM або serverless-ресурси з чутливими даними [2]. Через різні вибірки та знаменники ці значення не об'єднуються в одну статистичну сукупність.

Verizon DBIR 2026 використано для характеристики початкового вектора компрометації: 31% зафіксованих порушень починалися з експлуатації програмних вразливостей [26]. Unit 42 застосовано як галузеве підтвердження поширеності непропатчених HIGH/CRITICAL дефектів у виробничих кодових базах і публічних хмарних хостах [4]. Окремий клас ризику джерела образу підтверджує дослідження whoAMI: Datadog Security Labs описала можливість підміни AMI через нечіткий пошук за назвою, а документація AWS фіксує Allowed AMIs як account-level guardrail для дозволених образів [7–8]. Отже, ці джерела обґрунтовують різні елементи моделі: patch cadence, пріоритизацію вразливостей і перевірку походження образу.

Причини неналежного керування життєвим циклом машинних образів

Cloud Security Alliance «Top Threats to Cloud Computing 2024» використано для організаційного виміру ризику: за результатами опитування понад 500 фахівців неправильна конфігурація та неналежний контроль змін посіли перше місце серед хмарних загроз [6]. Звідси в моделі виведено потребу у визначеному власнику образу, шлязі схвалення, RACI, строках дії винятків і контрольованій депрекації. Технічними проявами цієї проблеми є ad-hoc створення AMI та шаблонів, drift між середовищами й відсутність сканування в конвеєрі.

Окремою типовою хвилю є припущення, що оновлення образу автоматично оновлює парк: запущені VM продовжують працювати зі старого цифрового відбитка до повторного розгортання. Звіти Sysdig використано лише як галузевий runtime-проху для оцінювання зрілості shift-left: вони зіставляють спрацювання політик у runtime та CI/CD і показують зміну частки використовуваних HIGH/CRITICAL вразливостей у 2024–2026 роках [5; 17–18]. Ці дані не трактуються як результат конвеєра еталонних VM-образів.

Матриця ризиків життєвого циклу машинних образів

Пріоритети в таблиці 1 визначено не за назвою вразливості й не лише за базовим CVSS. Відповідно до NIST SP 800-30 ризик розглядається як поєднання ймовірності реалізації загрози та масштабу шкоди для операцій, активів і місії [27]. Для технічних вразливостей оцінку уточнюють: тяжкість за CVSS v4.0 [28], прогноз ймовірності експлуатації EPSS [29], факт активної експлуатації у CISA KEV та рішення SSVC з урахуванням технічного впливу, автоматизованості атаки й поширеності в середовищі [30–31]. Для ризику машинного образу додатковим множником є радіус поширення: одна помилка базового артефакту успадковується всіма похідними VM. Тому оцінка є сценарною: базовий пріоритет підвищується за наведених умов ескалації, а залишковий рівень допускається лише після технічної перевірки контролів.



Таблиця 1.

Контекстна матриця пріоритизації ризиків життєвого циклу машинних образів

Ризиковий сценарій	Доказова підстава	Базовий пріоритет	Умови ескалації	Контролі та залишковий ризик
R1. Застарілі пакети або EOL-компоненти масово успадковуються з образу	Відомі CVE залишаються придатними для повторної експлуатації; CISA вимагає пріоритетного усунення KEV [31].	Високий	Критичний: KEV або SSVC «Act»; internet-facing VM; відсутній компенсувальний контроль; великий парк похідних VM.	Scheduled rebuild, KEV/EPSS/CVSS gate, EOL policy, депрекація та підтверджений rollout. Залишковий: середній; низький лише після заміни всіх залежних VM.
R2. Недовірене джерело або підміна образу	whoAMI продемонструвала виконання коду через помилковий вибір AMI; довіра до owner є окремим контролем [7–8].	Критичний	Зберігається критичним, якщо автоматизоване розгортання приймає public/community image без фіксованого owner, allowlist або перевірки підпису.	Allowed/Trusted Images, owner ID, digest/signature verification, deny-by-default. Залишковий: низький–середній.
R3. Секрети, ключі або токени потрапляють до образу	Секрет копіюється в кожен VM і може забезпечити доступ поза межами одного хоста; вплив залежить від чинності та привілеїв.	Високий	Критичний: чинний privileged/cloud credential, широке поширення образу, можливість lateral movement або доступ до production.	Secret scan, ephemeral credentials, scrub history, ротація після інциденту. Залишковий: середній; низький після ротації та перевидання образу.
R4. Небезпечна конфігурація базового образу	Помилка baseline відтворюється на всіх похідних VM; NIST оцінює імовірність разом із впливом на місію [27].	Високий	Критичний: вимкнені автентифікація, журналювання або захисні агенти на зовнішньо доступних чи критичних VM.	CIS/STIG baseline, configuration-as-code, negative tests, policy gate. Залишковий: середній.
R5. Підміна артефакту або компрометація builder	Компрометований конвеєр може створити довірений на вигляд образ; SLSA вимагає provenance та перевірки цілісності [25].	Критичний	Критичний за доступу до production publication/signing або за відсутності незалежної перевірки provenance.	Ізольований ephemeral builder, signing, provenance, RBAC, separation of duties. Залишковий: середній.



Ризиковий сценарій	Доказова підстава	Базовий пріоритет	Умови ескалації	Контролі та залишковий ризик
R6. Надмірні права на публікацію та спільний доступ	Надмірні дозволи дають змогу обійти approval і поширити неперевірений образ.	Високий	Критичний: одна роль одноосібно збирає, підписує й публікує образ у production або між обліковими записами.	Least privilege, SoD, MFA, approval, audit trail. Залишковий: низький–середній.
R7. Configuration drift після розгортання	Ручні зміни руйнують підтверджену baseline і зменшують достовірність результатів сканування образу.	Високий	Критичний: drift вимикає EDR, журналювання, контроль доступу або створює зовнішню експозицію.	Immutable replacement, drift detection, CMDB, заборона ручних змін. Залишковий: середній.
R8. Образ виправлено, але парк VM не замінено	Патч у каталозі не усуває вразливість у вже запущених VM; фактичний ризик існує до завершення rollout.	Високий	Критичний: KEV/SSVC «Акт», зовнішня експозиція, критичні дані або прострочений remediation SLA.	Inventory by image ID, age policy, staged replacement, rollout SLA та контроль покриття. Залишковий: середній до 100% заміни.
R9. Розрив відповідальності за rebuild, винятки та expiry	Відсутність owner і строку дії створює системну умову для накопичення всіх попередніх ризиків.	Середній	Високий: немає owner/RACI/SLA, винятки без expiry, прострочені образи продовжують запускатися.	Owner tags, RACI, ITSM SLA, expiry, evidence retention. Залишковий: низький за зрілого процесу.

Матриця показує, чому однакова позначка «середній» для всіх сценаріїв є некоректною. Підміна джерела та компрометація builder мають критичний базовий пріоритет через можливість масового виконання недовіреного коду. Ризики застарілих компонентів і незаміненого парку переходять із високого до критичного, коли присутня активна експлуатація, зовнішня доступність або значний радіус поширення. Водночас залишковий ризик не можна знижувати лише за фактом оновлення гесіре: потрібні докази нового сканування, допуску та заміни похідних VM.

Огляд ринку засобів захисту та незакрита прогалина

Таблиця 2.

Карта ринку та покриття ризиків

Рівень	Приклади засобів	Що перекриває	Незакрита прогалина
Build/reproducibility	Packer/HCP Packer; EC2/Azure Image Builder [9; 19–20]	Версіонований recipe, build/test	Сам build не гарантує production admission
Hardening	CIS Hardened Images/Benchmarks; Ansible [10]	Secure baseline, STIG/CIS mapping	Потрібні винятки й patch cadence
Scanning	Trivy/Harbor; Tenable; CNAPP/CWPP [23; 34]	CVE, secrets, SBOM, rescan	Не виконує rollout; пізні CVE DB



Рівень	Приклади засобів	Що перекриває	Незакрита прогалина
Gallery/lifecycle	AMI; Azure Compute Gallery; GCE image families; vSphere Content Library [19–22]	Versions, sharing, deprecation	Немає єдиного approval/evidence model
Admission guardrail	AWS Allowed AMIs; Azure Policy; GCP Trusted Images; VMware RBAC [7–8; 20–22]	Блокує недовірені джерела	Platform-specific правила
Supply chain	SLSA provenance; signing; SBOM [24–25]	Походження і цілісність	Нерівномірна підтримка VM image formats
Governance/audit	Cherwell/ServiceNow; CMDB; policy-as-code [15; 39–41]	Owner, SLA, approval, evidence	Переважно кастомна інтеграція
Runtime feedback	Sysdig; CrowdStrike; Splunk; CNAPP [5; 17–18]	Drift, threats, prioritization	Не виправляє base image без rebuild/redeploy

Наукові праці використано для обґрунтування окремих механізмів запропонованої системи. Систематичні та емпіричні дослідження DevSecOps і безперервного тестування визначають потребу раннього вбудовування контролів у CI/CD, автоматичного повторення перевірок і спільної відповідальності команд [11–12; 36–38]. Праці про security-as-code та policy-as-code підтверджують можливість формалізувати доступ і рішення про допуск як код [15–16; 39]. Дослідження IaC-оркестрації та захищених конвеєрів використано для вимог відтворюваності, журналювання доказів і розмежування повноважень [40–41]. Власні суміжні дослідження авторів охоплюють керування секретами та автоматизацію хмарних інцидентів [13–14], однак не розв'язують задачу керування походженням і заміною VM-образів. Саме відсутність цілісного багатоплатформного циклу «build — scan — policy gate — promotion — runtime feedback — replacement» визначає незакрити прогалину цієї роботи.

Незакрита прогалина полягає не у відсутності сканера чи галереї, а у відсутності єдиного vendor-neutral процесу, який одночасно забезпечує trusted source, відтворюваний build, policy gate, separation of duties, багатоплатформну промоцію, аудиторські докази, expiry/deprecation і заміну вже розгорнутих VM. Саме ця інтеграційна прогалина визначає науково-практичну новизну системи.

ЗАПРОПОНОВАНА АРХІТЕКТУРА АВТОМАТИЗОВАНОГО КОНВЕЄРА ЕТАЛОННИХ ОБРАЗІВ

Запропоноване рішення є інтегрованим конвеєром, що поєднує формування незмінних образів, їхнє ґартування, автоматизоване сканування, контрольоване схвалення та промоцію до промислових галерей чотирьох цільових платформ. Перший рисунок подає цілісний процес постачання, а другий — компактне архітектурне бачення системи; детальні функціональні контури наведено на рис. 2.1–2.6.

Процес постачання на рис. 1 складається з чотирьох послідовних кроків. На кроці 1 інженер DevSecOps формує запит, указує версію та обирає цільову платформу. На кроці 2 Jenkins запускає Packer/Ansible-збірку, що формує платформний кандидат образу. На кроці 3 кандидат розгортається в ізольованій sandbox-галереї, де виконуються функціональні тести, сканування вразливостей і перевірка доказів походження. На кроці 4 InfoSec фіксує рішення: результат PASS дозволяє промоцію до промислової галереї, а BLOCK повертає артефакт на виправлення та повторне складання.

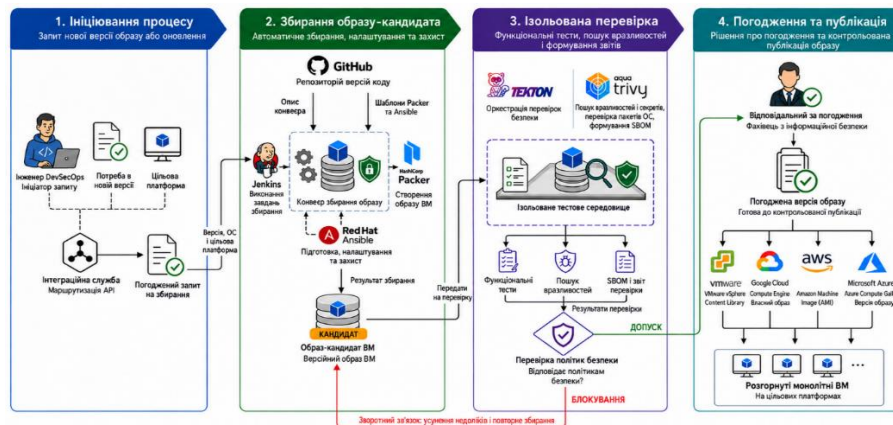


Рис. 1. Цілісний процес формування, перевірки та промоції еталонного образу

Таким чином, результатом процесу є не лише версія VM-образу, а й пов'язаний набір доказів: ідентифікатор збірки, журнал тестів, звіт сканування та рішення про допуск. Негативний результат не створює частково погодженого артефакту, а однозначно повертає збірку до попереднього кроку.

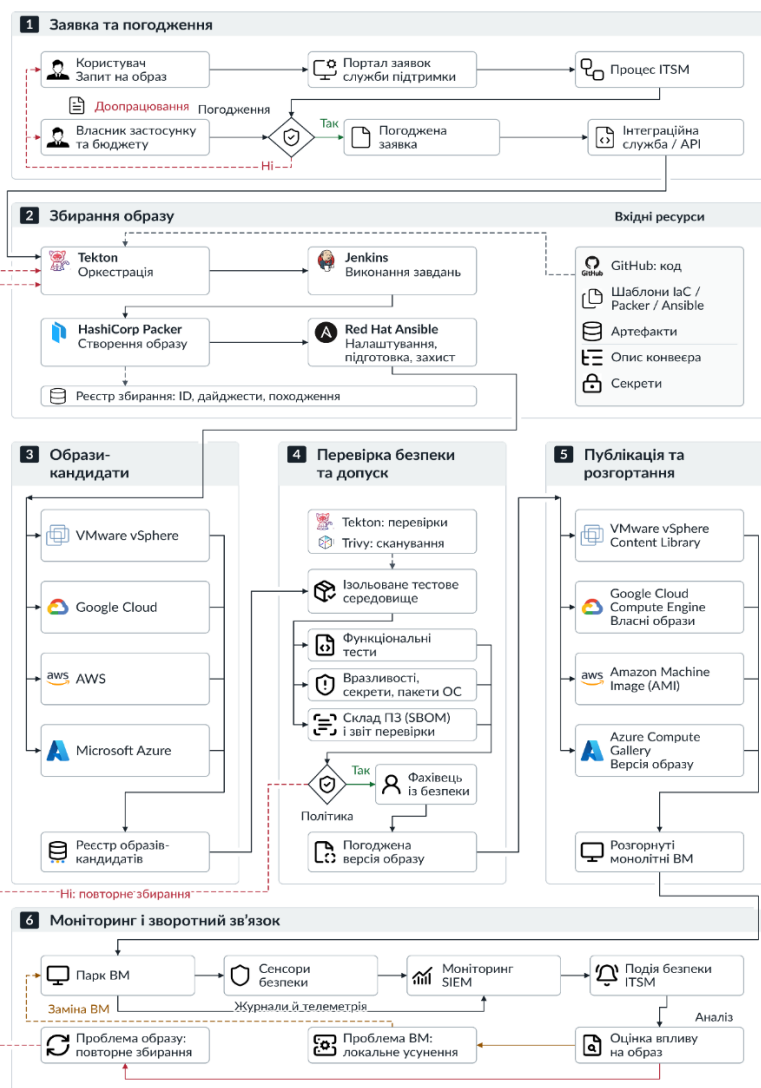


Рис. 2. Архітектурне бачення конвеєра еталонних образів для монолітних VM-навантажень

Архітектурне бачення на рис. 2 зводить ці кроки в єдиний контур: погоджений запит надходить через узагальнений інтеграційний сервіс до Tekton/Jenkins; GitHub постачає визначення pipeline і шаблони; Packer/Ansible формують кандидатні образи; Trivy та шлюз політик приймають технічне рішення; погоджена версія публікується у нативних каталогах платформ і розгортається як монолітний VM-парк. Дані runtime-моніторингу повертають image-level проблеми до ITSM і повторного складання.

Рисунок 2 відокремлює об'єкти дослідження від допоміжних корпоративних сервісів. Packer, Ansible, Jenkins, Tekton і Trivy показано як технологічне ядро експериментального конвеєра; інтеграція, ITSM, каталоги платформ і runtime-моніторинг подані як логічні функції. Таке подання не створює хибного враження, що реалізація залежить від конкретного великого інтеграційного продукту.

На відміну від рисунка 1, який відображає послідовність постачання образу, рисунок 2 показує внутрішню будову системи, розподіл відповідальності між її функціональними контурами та два різновиди зворотного зв'язку: локальне усунення проблеми окремої VM і повторне складання еталонного образу за виявлення дефекту на рівні базового артефакту.

Візуальні позначення у схемі застосовано послідовно: суцільні чорні стрілки показують основний рух артефактів і керуючих рішень, зелений маршрут — допуск до публікації, червоний — повернення образу на повторне складання, а помаранчевий — заміну або локальне усунення проблеми окремої VM. Офіційні назви Packer, Ansible, Jenkins, Tekton і Trivy залишено без перекладу, тоді як функціональні ролі, стани та результати подано українською мовою. Це дає змогу читати рисунок як єдину систему, не ототожнюючи логічні функції з конкретним корпоративним продуктом.

Деталізація функціональних контурів

Наведені нижче шість схем деталізують функціональні контури архітектури в єдиному візуальному стилі. Кожний рисунок описує завершений етап, його входи, керуючі компоненти, контрольні рішення та результати; переходи між рисунками позначено спільними граничними артефактами — схваленим запитом на складання, кандидатними VM-образами, погодженою версією образу та парком активних VM.

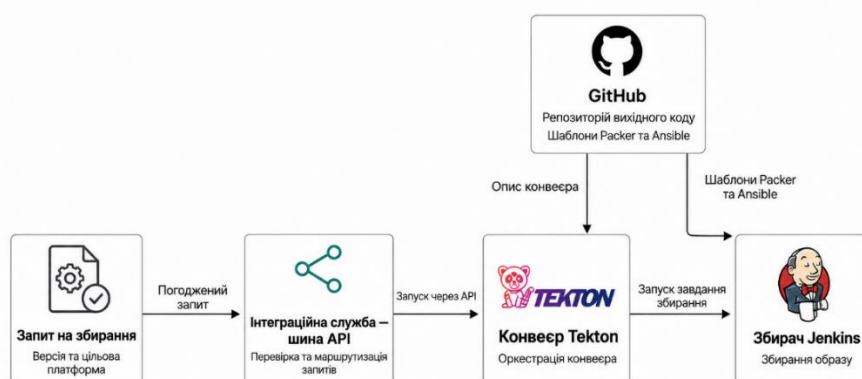


Рис. 2.1. Ініціювання та оркестрація процесу формування еталонного образу віртуальної машини

Погоджений Build Request надходить через інтеграційний сервіс до Tekton, який оркеструє процес і запускає build job у Jenkins. GitHub окремо постачає Tekton визначення pipeline, а Jenkins — Packer/Ansible-шаблони. Такий поділ відокремлює бізнес-ініціювання, керування процесом, зберігання коду та безпосереднє виконання збірки, забезпечуючи простежуваність версії, цільової платформи й параметрів запуску.

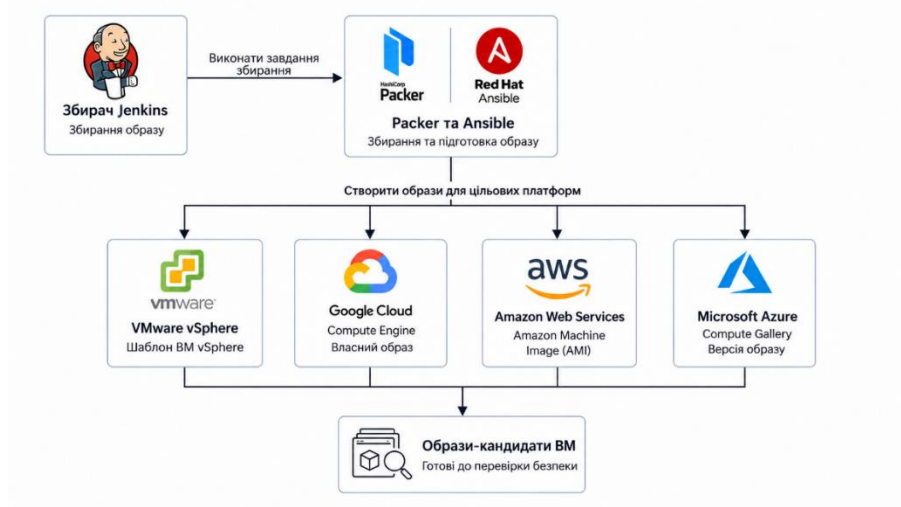


Рис. 2.2. Багатоплатформне формування кандидатних образів віртуальних машин

Jenkins запускає узгоджений процес, у якому Packer формує платформні образи, а Ansible встановлює пакети, застосовує ґартування й налаштовує агенти. Результатом є чотири окремі кандидати: vSphere VM Template, Google Compute Engine Custom Image, AWS AMI та Azure Compute Gallery Version. На цьому етапі вони ще не вважаються погодженими й передаються на перевірку безпеки. Експериментальна перевірка показала, що після full-upgrade кандидат має перезавантажитися в нове ядро, пройти dry-run видалення застарілих versioned kernel packages і повторний boot-test до передачі на сканування.

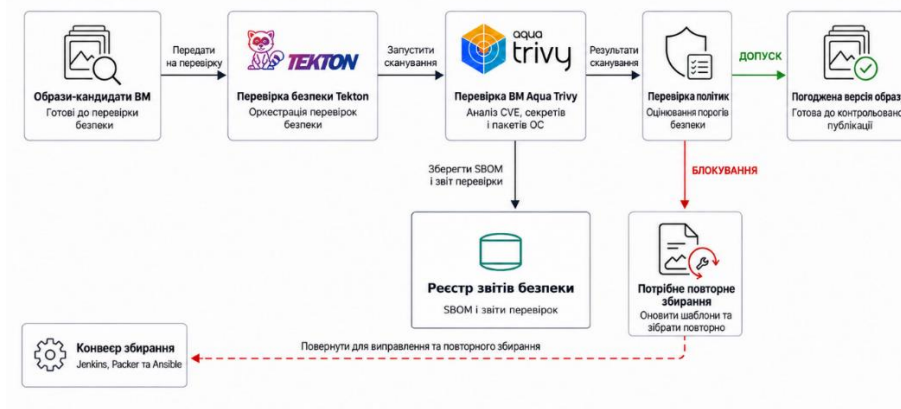


Рис. 2.3. Автоматизоване сканування та перевірка кандидатних образів за політиками безпеки

Tekton оркеструє перевірку безпеки, а Trivy аналізує CVE, секрети та пакети ОС і формує SBOM та звіт сканування. Policy Gate порівнює результати з установленими порогоми: PASS створює Approved Image Version, а BLOCK повертає проблему до Jenkins, Packer і Ansible для виправлення та повторного складання. Звіт, SBOM і рішення про допуск зберігаються в узагальненому реєстрі доказів перевірки.

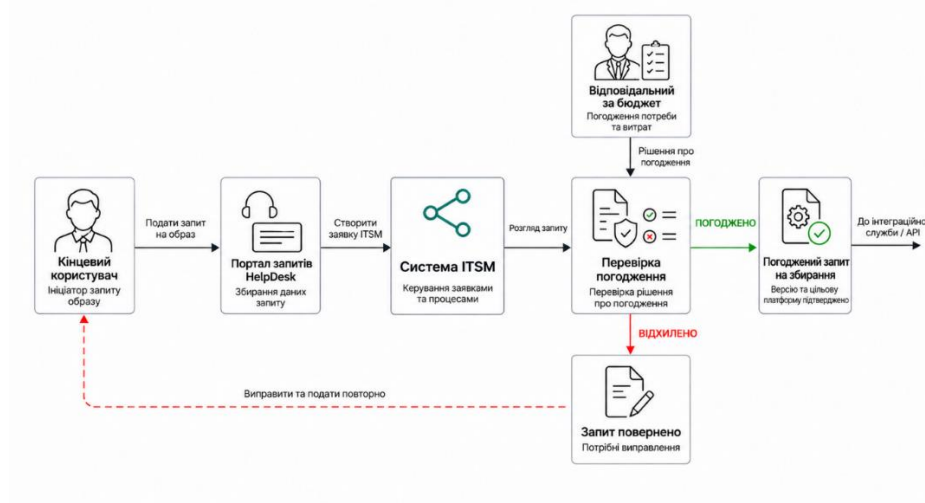


Рис. 2.4. Реєстрація та погодження запиту на формування еталонного образу в ITSM-системі

Користувач подає запит через HelpDesk, після чого ITSM-система реєструє тикет, стан workflow та аудиторський слід. Власник бюджету надає бізнесове й фінансове рішення. Схвалений запит перетворюється на Approved Build Request і передається до інтеграційного сервісу; відхилений або неповний запит повертається заявнику для уточнення. Це організаційне погодження не замінює подальший security gate.

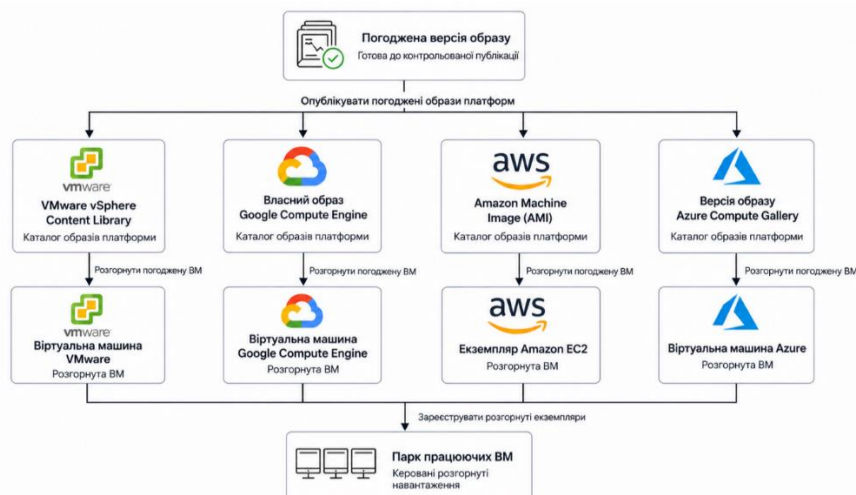


Рис. 2.5. Публікація погоджених образів і розгортання віртуальних машин у багатоплатформному середовищі

Погоджені платформні артефакти публікуються у відповідних нативних каталогах: vSphere Content Library, Google Custom Images, AWS AMI та Azure Compute Gallery. Кожний каталог створює лише сумісний із ним тип VM. Ідентифікатори розгорнутих екземплярів об'єднуються у керований Runtime VM Fleet, що дає змогу контролювати версію джерельного образу, його вік і покриття схваленими версіями.

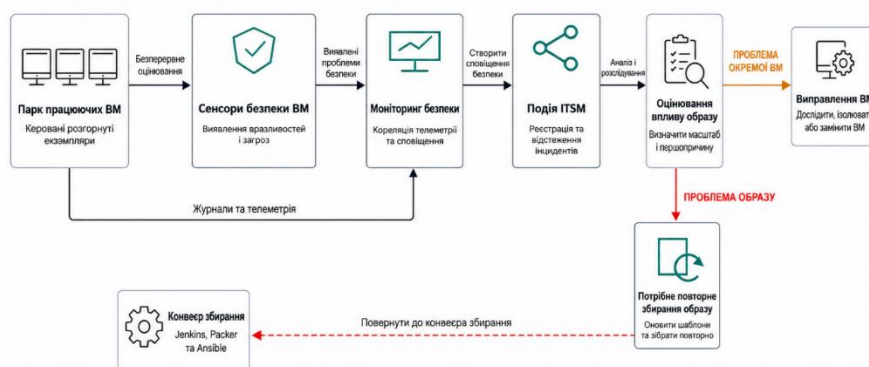


Рис. 2.6. Моніторинг безпеки віртуальних машин і зворотний зв'язок із процесом формування еталонного образу

Runtime VM Fleet передає журнали й телеметрію до системи моніторингу, а сенсори безпеки додають результати vulnerability та endpoint-контролю. ITSM реєструє подію, після чого Image Impact Assessment відокремлює локальний інцидент однієї VM від дефекту базового образу. Лише image-level issue повертається до build pipeline; instance-specific issue усувається локально без необґрунтованого rebuild усього еталонного образу.

Зіставлення компонентів архітектури з еталонними практиками

Кожен компонент запропонованої архітектури відповідає опублікованому еталонному патерну, що підтверджує обґрунтованість рішення. Це зіставлення узагальнено в таблиці 3.

Таблиця 3

Зіставлення компонентів конвеєра еталонних образів з еталонними практиками

Компонент / етап	Реалізація у запропонованій системі	Еталонна практика (джерело)
Складання незмінного образу	HashiCorp Packer + Ansible формують образ як версіонований код для чотирьох платформ	Packer / HCP Packer як стандарт мультиплатформного складання з відкликанням застарілих образів (HashiCorp)
Гартування конфігурації	Ansible застосовує базові лінії гартування до образу	CIS Hardened Images / CIS Benchmarks (Center for Internet Security)
Сканування у конвеєрі	Trivy сканує образ до публікації; збірка провалюється за наявності критичних вразливостей понад установленний поріг	Scan-and-fail під час складання; Trivy як типовий open-source сканер образів (Aqua Security)
Реєстр артефактів	Реєстр доказів зберігає звіт сканування, SBOM, ідентифікатор збірки та рішення про допуск	Версіонована provenance та цілісність артефактів відповідно до SLSA і NIST SSDF
Шлюз схвалення InfoSec	Валідація SIEM/AV/Vuln-Mgmt; політика-як-код провалює невідповідні образи; промоція лише після підтвердження	Політика-як-код і блокування нееталонних образів контролерами допуску (Wiz)
Промоція та вивід з експлуатації	Шлюз ITSM зі схваленням відповідального власника; депрекація застарілих версій образів	Промоційний шлях dev → test → prod із контрольованим виведенням образів з експлуатації



Таким чином, запропонована архітектура не є ізольованим внутрішнім проектом, а узгоджується з опублікованими еталонними патернами провідних постачальників, що підвищує її науково-практичну обґрунтованість та відтворюваність.

МЕТОДИКА ОЦІНЮВАННЯ ТА АНАЛІТИЧНІ РЕЗУЛЬТАТИ

Методологія дослідження та джерела даних

Дослідження поєднує архітектурне проектування з контрольованим локальним експериментом. Стенд розгорнуто на Debian з QEMU/KVM і libvirt. Чотири історичні Ubuntu 24.04 LTS cloud images від 23.05.2024, 19.11.2024, 16.05.2025 і 13.12.2025 було перевірено за SHA-256 та GPG-підписами Canonical. Для treatment-збірок використано один підписаний Ubuntu Snapshot Service з timestamp 20260930T000000Z [32–33].

Образи склалися Packer 1.16.1 [35] з Ansible 12.0.0. Усі варіанти мали однаковий Nginx-профіль, прикладне навантаження, 2 vCPU, 2560 MiB RAM та 20 GiB віртуального диска. Сканування виконувалося Trivy 0.74.0 з однією замороженою базою та з параметром --skip-db-update [34]. QCOW2 монтувалися лише для читання; збіг SHA-256 до і після сканування підтвердив незмінність усіх артефактів.

Протокол вимірювання рівня вразливостей

Перевірялося припущення H1, за яким відтворюваний конвеєр еталонних образів зменшує кількість успадкованих виправних HIGH/CRITICAL CVE та підвищує покриття hardening-контролів без порушення функціональності. Кожна пара порівнювала три стани: baseline; combined-delayed після full-upgrade та hardening; combined-cleanup після додаткового перезавантаження в нове ядро та точного видалення застарілих versioned kernel packages.

Основною метрикою була кількість унікальних HIGH і CRITICAL CVE з непорожнім FixedVersion. Покриття ризику обчислювалося за формулою:

$$C = \frac{V_{\text{before}} - V_{\text{after}}}{V_{\text{before}}} \times 100\%$$

Додатково фіксувалися унікальні CVE за рівнями, CycloneDX SBOM, secret findings, 11 hardening-контролів, 5 функціональних тестів, HTTP-статус та рішення policy gate. PASS, REVIEW і BLOCK мали коди 0, 10 і 20 відповідно; наявність виправної CRITICAL CVE спричиняла BLOCK.

Експериментальні результати

Агреговані результати наведено в таблиці 4. Звичайний full-upgrade зменшив основну метрику лише на 75 findings. Причиною була присутність в файловій системі старих пакетів ядра, хоча runtime вже використовував 6.8.0-142-generic.

Таблиця 4.

Агреговані результати контрольованого експерименту

Показник	Baseline	Після full-upgrade	Після kernel cleanup
Fixable HIGH	1587	1512	0
Fixable CRITICAL	93	93	0
Fixable HIGH+CRITICAL	1680	1605	0
Покриття основної метрики	—	4,46%	100%
Hardening-контролі	9/11 (81,82%)	11/11 (100%)	11/11 (100%)
Функціональні тести	5/5	5/5	5/5
Policy gate	BLOCK	BLOCK	REVIEW

Повний керований цикл додав два важливі кроки: перезавантаження build-VM у нове ядро та видалення лише точно визначених застарілих kernel packages після dry-run. Видалено 22 пакети; активне ядро, linux-virtual, linux-image-virtual і linux-headers-virtual збережено. Після cleanup виправні HIGH/CRITICAL зменшилися до нуля, а всі чотири рішення policy gate змінилися з BLOCK на REVIEW. У кожному cleanup-образі залишилося 162 HIGH і 5 CRITICAL CVE без доступного FixedVersion. Вони не враховувалися як усунуті, а policy gate обґрунтовано зберіг REVIEW. Усі чотири VM пройшли 5/5



функціональних, 11/11 hardening- і 5/5 cleanup-integrity тестів, повернули HTTP 200 та штатно вимкнулися.

Таблиця 5.

Зміна виправних HIGH/CRITICAL CVE за експериментальними блоками

Дата source image	Baseline	Після upgrade	Після cleanup	Видалено kernel packages	Policy
23.05.2024	473	451	0	4	BLOCK → REVIEW
19.11.2024	449	427	0	6	BLOCK → REVIEW
16.05.2025	401	383	0	6	BLOCK → REVIEW
13.12.2025	357	344	0	6	BLOCK → REVIEW
Разом	1680	1605	0	22	4 × BLOCK → REVIEW

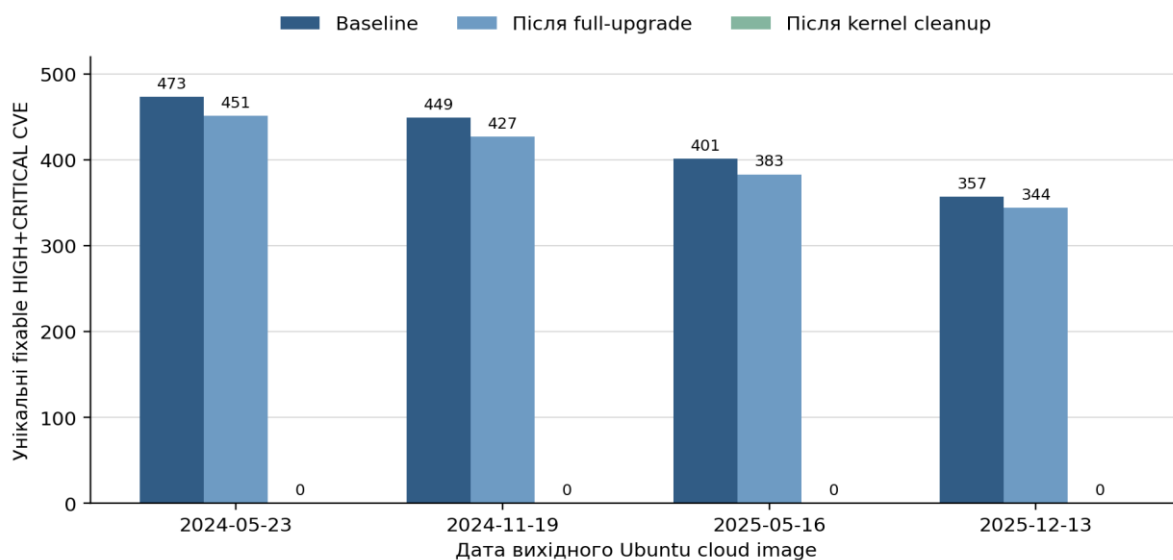


Рис. 3. Зміна кількості виправних HIGH/CRITICAL CVE на трьох етапах обробки образу

Видалення застарілих kernel packages дало однаково позитивний результат у всіх чотирьох парах. Точний двосторонній sign test для $n=4$ дав $p=0,125$; отже, спостереження є стабільним в межах пілоту, але не дає підстав для твердження про статистичну значущість.

Перевірена ринкова статистика 2024–2026 років

Таблиця 6.

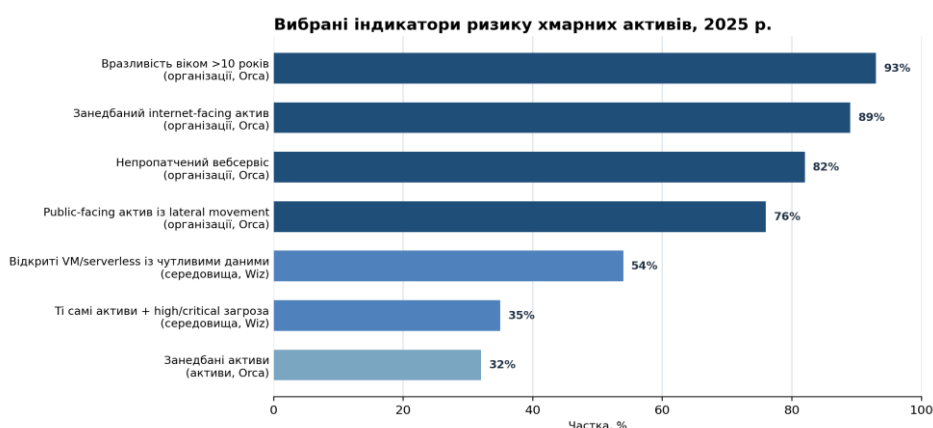
Перевірені галузеві показники, релевантні керуванню образами

Джерело	Перевірений показник	Інтерпретація
Orca, 2025	115 CVE на середній актив; 32% активів neglected; 76% організацій мають public-facing актив із lateral movement	Масштаб проблеми; lifecycle control
Orca, 2025	93% організацій мають CVE віком >10 років; 82% — непропатчений web service; 89% — internet-facing neglected asset	Регулярний rebuild і retirement



Джерело	Перевірений показник	Інтерпретація
Wiz, 2025	54% середовищ мають відкриті VM/serverless із sensitive data; 35% — також high/critical threat	Пріоритизація exposure + data + CVE
Verizon DBIR, 2026	31% порушень безпеки починалися з експлуатації програмних вразливостей; цей спосіб випередив викрадені облікові дані	Актуальність patch cadence і превентивного контролю
Sysdig, 2024	91% policy failure у runtime проти 71% у CI/CD; high/critical in-use знижено майже на 50% p/p	Shift-left ще неповний
Sysdig, 2025	In-use vulnerabilities <6%; поліпшення на 64% за два роки	Зіставний проху динаміки, не golden-image експеримент
Sysdig, 2026	>70% команд застосовують behavior detection у 91% середовищ; auto-kill adoption +140% p/p	Runtime feedback та response automation

Показники мають різні вибірки та знаменники; їх не можна підсумовувати або подавати як єдиний часовий ряд [1–5; 17–18; 26].



Примітка: показники Orca Security та Wiz [1–2] мають різні знаменники (організації, середовища, активи), тому не утворюють єдиної вибірки й подані як карта ризиків, а не рейтинг.

Рис. 4. Вибрані індикатори ризику хмарних активів за даними Orca Security та Wiz, 2025 р.

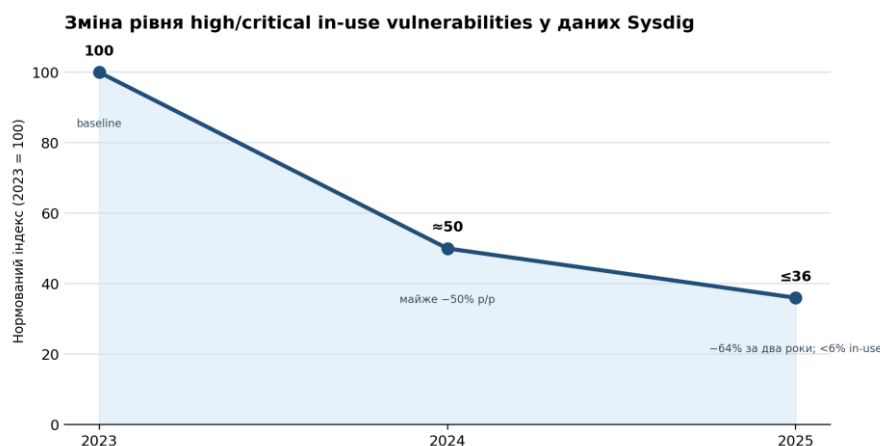


Рис. 5. Нормована галузева динаміка high/critical in-use vulnerabilities, 2023–2025 pp.



Індекс обчисленого з опублікованих відносних змін Sysdig: 2024 р. – майже 50% зниження рік до року; 2025 р. – 64% поліпшення за два роки. Це галузевий проху для runtime, а не результат golden-image експерименту [3–4]. Розраховано за опублікованими відносними змінами Sysdig [5; 17]. Це галузевий runtime проху, а не результат конвеєра еталонних образів.

Зіставлення з галузовими орієнтирами

Галузеві звіти використано не для підміни власного експерименту, а для зіставлення його механізму з ширшим контекстом. Спостережене в пілоті 100% покриття стосується лише виправних HIGH/CRITICAL CVE в чотирьох парах образів. Воно не може бути прямо зіставлене з телеметрією Oгса, Wiz або Sysdig через різні вибірки та знаменники.

Отримані дані уточнюють механізм ризику: оновлення встановлює нове ядро, але без контрольованого cleanup застарілі пакети залишаються частиною образу та підтримують його статус BLOCK.

Контрольований експеримент підтвердив, що звичайний full-upgrade не є достатнім кроком для формування еталонного образу. Після оновлення старі пакети ядра залишалися в образі та пояснювали всі 1605 виправних HIGH/CRITICAL findings, що залишалися. Перезавантаження в нове ядро, dry-run та точне видалення 22 застарілих пакетів усунули цей клас ризику.

Практичне значення policy gate полягає не лише в автоматичному блокуванні, а й у коректному розмежуванні виправного та залишкового ризику. Після cleanup виправні CRITICAL зникли, тому BLOCK було знято. Проте п'ять CRITICAL без FixedVersion залишилися в evidence, тому образ перейшов до REVIEW, а не PASS. Така поведінка відповідає моделі контрольованого допуску, а не приховування не виправних CVE.

Обмеження дослідження. Пілот охоплює чотири пари Ubuntu 24.04/Nginx, один APT snapshot, Trivy 0.74.0 та одну заморожену CVE-базу. Точний sign test дав $p=0,125$; тому статистична значущість і популяційна причинність не заявляються. Локальний стенд перевіряв артефакт і runtime-поведінку, але не публікацію в AWS, Azure, GCP чи vSphere. MTTR, drift парку та частка VM з approved image не вимірювалися. Нуль secret findings означає лише відсутність знахідок за застосованим ruleset.

Перспективи подальших досліджень. Потрібно збільшити кількість незалежних пар, повторити протокол для інших ОС і ролей, а також перевірити публікацію та заміну VM у реальних хмарних каталогах. Окремими напрямками є криптографічне підписування, SLSA/SBOM attestation, вимірювання MTTR і image drift, а також дослідження контейнерних і serverless-артефактів.

ВИСНОВКИ

У роботі розглянуто актуальну та недостатньо висвітлену проблему безпеки публічної хмари — розрив між захистом окремого екземпляра та керуванням життєвим циклом машинних образів, з яких ці екземпляри створюються. Показано, що за відсутності централізованого процесу до промислового середовища потрапляють застарілі й вразливі еталонні образи, а похідні монолітні VM-навантаження успадковують їхні дефекти.

Запропоновано референсну архітектуру автоматизованого конвеєра еталонних образів для чотирьох цільових платформ — VMware vSphere, GCE Custom Images, AWS AMI та Azure Compute Gallery. Технологічне ядро використовує HashiCorp Packer, Ansible, Jenkins/Tekton і сканування Trivy, тоді як інтеграція, ITSM, моніторинг і реєстр доказів описані як логічні функції без прив'язки до конкретного вендора.

Контрольований експеримент з чотирма історичними Ubuntu cloud images показав, що full-upgrade сам по собі зменшив 1680 виправних HIGH/CRITICAL CVE до 1605, тобто забезпечив 4,46% покриття. Після перезавантаження в нове ядро та видалення 22 застарілих kernel packages кількість виправних HIGH/CRITICAL зменшилася до нуля. Усі збірки зберегли перевірену функціональність, а policy gate змінив рішення з BLOCK на REVIEW.

Науковий внесок роботи полягає в обґрунтуванні керування життєвим циклом еталонних образів як окремого рівня контролю безпеки для монолітних VM-навантажень і в експериментальному виокремленні ролі очищення застарілих пакетів. Отриманий 100% показник стосується лише спостережених виправних HIGH/CRITICAL CVE за умов пілоту; $n=4$ та $p=0,125$ не дають підстав для загальної статистичної значущості.



СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Orca Security. (2025). 2025 state of cloud security report. <https://orca.security/resources/state-of-cloud-security/>
2. Wiz. (2025). Cloud data security snapshot 2025. <https://www.wiz.io/reports/cloud-data-security-report-2025>
3. IBM Security, & Ponemon Institute. (2025). Cost of a data breach report 2025. IBM. <https://www.ibm.com/reports/data-breach>
4. Palo Alto Networks, Unit 42. (2023). Cloud threat report: Volume 7. <https://unit42.paloaltonetworks.com/cloud-threat-report/>
5. Sysdig. (2025). 2025 cloud-native security and usage report. <https://www.sysdig.com/resources/reports/2025-cloud-native-security-and-usage-report>
6. Cloud Security Alliance. (2024). Top threats to cloud computing 2024. <https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2024>
7. Datadog Security Labs. (2025). whoAMI: A cloud image name confusion attack. <https://securitylabs.datadoghq.com/articles/whoami-a-cloud-image-name-confusion-attack/>
8. Amazon Web Services. (2024). Allowed AMIs: Account-level guardrail for approved Amazon Machine Images. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-allowed-amis.html>
9. HashiCorp. (2024). Build a golden image pipeline with HCP Packer. HashiCorp Developer. <https://developer.hashicorp.com/packer/tutorials/cloud-production/golden-image-with-hcp-packer>
10. Center for Internet Security. (2024). CIS Hardened Images and CIS Benchmarks. <https://www.cisecurity.org/cis-hardened-images>
11. Mohammed, K. I., Shanmugam, B., & El-Den, J. (2025). Evolution of DevSecOps and its influence on application security: A systematic literature review. *Technologies*, 13(12), Article 548. <https://doi.org/10.3390/technologies13120548>
12. Rangnau, T., van Buijtenen, R., Fransen, F., & Turkmen, F. (2020). Continuous security testing: A case study on integrating dynamic security testing tools in CI/CD pipelines. In *Proceedings of the 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 145–154). <https://doi.org/10.1109/EDOC49727.2020.00026>
13. Chornii, V., Martseniuk, Y., Partyka, A., & Harasymchuk, O. (2025). Information security risks associated with the uncontrolled storage of secrets in source code. *CEUR Workshop Proceedings*, 4042, 250–271. <https://ceur-ws.org/Vol-4042/>
14. Sapsai, O., Martseniuk, Y., Partyka, A., & Harasymchuk, O. (2025). Research on automated security incident management in public cloud environments. *CEUR Workshop Proceedings*, 4042, 226–249. <https://ceur-ws.org/Vol-4042/>
15. Vakhula, O., Opirskyy, I., Vorobets, P., Bobko, O., & Kulinich, O. (2025). Research on policy-as-code for implementation of role-based and attribute-based access control. *CEUR Workshop Proceedings*, 3991, 139–157. <https://ceur-ws.org/Vol-3991/>
16. Vakhula, O., & Opirskyy, I. (2024). Research on security-as-code approach for cloud-native applications based on Kubernetes clusters. *CEUR Workshop Proceedings*, 3800, 58–69. <https://ceur-ws.org/Vol-3800/>
17. Sysdig. (2024). 2024 cloud-native security and usage report. <https://www.sysdig.com/2024-cloud-native-security-and-usage-report>
18. Sysdig. (2026). 2026 cloud-native security and usage report. <https://www.sysdig.com/press-releases/2026-usage-report>
19. Amazon Web Services. (n.d.). What is EC2 Image Builder? AWS Documentation. <https://docs.aws.amazon.com/imagebuilder/latest/userguide/what-is-image-builder.html>
20. Microsoft. (n.d.). Azure VM Image Builder overview. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/virtual-machines/image-builder-overview>
21. Google Cloud. (n.d.). Setting up trusted image policies. Google Cloud Documentation. <https://cloud.google.com/compute/docs/images/restricting-image-access>
22. Broadcom. (n.d.). Manage VM templates in a content library. VMware vSphere 8.0 Documentation. <https://techdocs.broadcom.com/us/en/vmware-cis/vsphere/vsphere-8-0/vsphere-virtual-machine-administration/using-content-libraries/vsphere-vm-admin/manage-vm-templates-in-a-content-library/vsphere-vm-admin.html>
23. Harbor. (2025). Vulnerability scanning (Harbor 2.13.0 documentation). <https://goharbor.io/docs/2.13.0/administration/vulnerability-scanning/>
24. National Institute of Standards and Technology. (2022). Secure Software Development Framework (SSDF), version 1.1 (NIST SP 800-218). <https://doi.org/10.6028/NIST.SP.800-218>



25. Supply-chain Levels for Software Artifacts. (2026). SLSA specification, version 1.2. <https://slsa.dev/spec/v1.2/>
26. Verizon. (2026). 2026 data breach investigations report. Verizon Business. <https://www.verizon.com/business/resources/reports/dbir/>
27. National Institute of Standards and Technology. (2012). Guide for conducting risk assessments (NIST SP 800-30 Rev. 1). <https://doi.org/10.6028/NIST.SP.800-30r1>
28. FIRST. (2023). Common Vulnerability Scoring System version 4.0: Specification document. <https://www.first.org/cvss/v4.0/specification-document>
29. FIRST. (n.d.). Exploit Prediction Scoring System (EPSS): Model and documentation. <https://www.first.org/epss/>
30. Cybersecurity and Infrastructure Security Agency. (n.d.). Stakeholder-Specific Vulnerability Categorization (SSVC). <https://www.cisa.gov/resources-tools/resources/stakeholder-specific-vulnerability-categorization-ssvc>
31. Cybersecurity and Infrastructure Security Agency. (n.d.). Known Exploited Vulnerabilities catalog. <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>
32. Canonical. (n.d.). Ubuntu cloud images. <https://cloud-images.ubuntu.com/>
33. Canonical. (n.d.). Ubuntu Snapshot Service. <https://snapshot.ubuntu.com/>; <https://ubuntu.com/server/docs/how-to/software/snapshot-service/>
34. Aqua Security. (n.d.). Trivy documentation. <https://trivy.dev/docs/>
35. HashiCorp. (n.d.). QEMU builder for Packer. HashiCorp Developer. <https://developer.hashicorp.com/packer/integrations/hashicorp/qemu/latest/components/builder/qemu>
36. War, A., Diallo, A., Habib, A., Klein, J., & Bissyandé, T. F. (2025). Vulnerabilities in infrastructure as code: What, how many, and who? *Empirical Software Engineering*, 30(5), Article 120. <https://doi.org/10.1007/s10664-025-10672-8>
37. Kumar, R., & Goyal, R. (2020). Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). *Computers & Security*, 97, Article 101967. <https://doi.org/10.1016/j.cose.2020.101967>
38. Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology*, 141, Article 106700. <https://doi.org/10.1016/j.infsof.2021.106700>
39. Paul, A., Manoj, R., & Udhayakumar, S. (2024). Amazon Web Services cloud compliance automation with Open Policy Agent. In *Proceedings of the 2024 International Conference on Expert Clouds and Applications (ICOECA)* (pp. 313–317). <https://doi.org/10.1109/ICOECA62351.2024.00063>
40. Diaz-De-Arcaya, J., Osaba, E., Benguria, G., Etxaniz, I., Lobo, J. L., Alonso, J., Torre-Bastida, A. I., & Almeida, A. (2023). IEM: A unified lifecycle orchestrator for multilingual IaC deployments. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering* (pp. 195–199). <https://doi.org/10.1145/3578245.3584938>
41. Morales, J. A., Hamed, J., Reynolds, D., Shepard, D., Antunes, L., Yankel, J., & Yasar, H. (2023). Experiences with secure pipelines in highly regulated environments. In *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES 2023)*, Article 57. <https://doi.org/10.1145/3600160.3605466>

**Yevhenii Martseniuk**

PhD Student, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0009-2289-0968
yevhenii.v.martseniuk@lpnu.ua

Bohdan Mankovskyi

Bachelor's Student, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0001-3857-9842
bohdan.mankovskyi.kb.2023@lpnu.ua

Oleh Deineka

M.Sc., Postgraduate Student, Department of Information Security
Lviv Polytechnic National University, Lviv, Ukraine
ORCID: 0009-0005-9156-3339
deinekaoleh.86@gmail.com

**AN AUTOMATED GOLDEN-IMAGE LIFECYCLE MANAGEMENT SYSTEM AS A MEANS OF
REDUCING PUBLIC-CLOUD INFRASTRUCTURE VULNERABILITIES**

Abstract. A large share of public-cloud security research and practice focuses on individual runtime instances, while organizational processes for managing machine images remain neglected. This study treats golden-image lifecycle management as a distinct security-control layer for monolithic applications deployed on virtual machines across VMware vSphere, Google Compute Engine, AWS, and Microsoft Azure. Containerized microservice delivery and Kubernetes admission controls are outside the primary scope because they use a different deployment artifact and lifecycle. The scientific novelty of this work lies in integrating an immutable image-build pipeline based on HashiCorp Packer and Ansible with Jenkins/Tekton orchestration, automated vulnerability scanning, a policy gate, evidence storage, runtime monitoring, and an ITSM approval workflow. Supporting corporate services are described as interchangeable logical functions rather than research targets, reducing unnecessary vendor coupling. The paper presents an integrated provisioning diagram, a compact architectural vision, six detailed stage diagrams, and a controlled local experiment with four historical Ubuntu 24.04 cloud images. Under one signed APT snapshot and one frozen Trivy database, ordinary full-upgrade reduced unique fixable HIGH/CRITICAL CVEs from 1,680 to 1,605, corresponding to 4.46% coverage. After boot verification and exact removal of 22 obsolete versioned kernel packages, the observed fixable HIGH/CRITICAL count fell to zero while all functional and hardening tests passed. Policy decisions changed from BLOCK to REVIEW because five CRITICAL CVEs without an available fixed version remained in every cleaned image.

Keywords: public cloud; golden image; machine image; immutable infrastructure; DevSecOps; HashiCorp Packer; vulnerability scanning; AWS AMI; Azure Compute Gallery; GCE Custom Images; VMware vSphere; image governance; shift-left.

REFERENCES

1. Orca Security. (2025). 2025 state of cloud security report. <https://orca.security/resources/state-of-cloud-security/>
2. Wiz. (2025). Cloud data security snapshot 2025. <https://www.wiz.io/reports/cloud-data-security-report-2025>
3. IBM Security, & Ponemon Institute. (2025). Cost of a data breach report 2025. IBM. <https://www.ibm.com/reports/data-breach>
4. Palo Alto Networks, Unit 42. (2023). Cloud threat report: Volume 7. <https://unit42.paloaltonetworks.com/cloud-threat-report/>
5. Sysdig. (2025). 2025 cloud-native security and usage report. <https://www.sysdig.com/resources/reports/2025-cloud-native-security-and-usage-report>
6. Cloud Security Alliance. (2024). Top threats to cloud computing 2024. <https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2024>



7. Datadog Security Labs. (2025). whoAMI: A cloud image name confusion attack. <https://securitylabs.datadoghq.com/articles/whoami-a-cloud-image-name-confusion-attack/>
8. Amazon Web Services. (2024). Allowed AMIs: Account-level guardrail for approved Amazon Machine Images. <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-allowed-amis.html>
9. HashiCorp. (2024). Build a golden image pipeline with HCP Packer. HashiCorp Developer. <https://developer.hashicorp.com/packer/tutorials/cloud-production/golden-image-with-hcp-packer>
10. Center for Internet Security. (2024). CIS Hardened Images and CIS Benchmarks. <https://www.cisecurity.org/cis-hardened-images>
11. Mohammed, K. I., Shanmugam, B., & El-Den, J. (2025). Evolution of DevSecOps and its influence on application security: A systematic literature review. *Technologies*, 13(12), Article 548. <https://doi.org/10.3390/technologies13120548>
12. Rangnau, T., van Buijtenen, R., Fransen, F., & Turkmen, F. (2020). Continuous security testing: A case study on integrating dynamic security testing tools in CI/CD pipelines. In *Proceedings of the 2020 IEEE 24th International Enterprise Distributed Object Computing Conference (EDOC)* (pp. 145–154). <https://doi.org/10.1109/EDOC49727.2020.00026>
13. Chornii, V., Martseniuk, Y., Partyka, A., & Harasymchuk, O. (2025). Information security risks associated with the uncontrolled storage of secrets in source code. *CEUR Workshop Proceedings*, 4042, 250–271. <https://ceur-ws.org/Vol-4042/>
14. Sapsai, O., Martseniuk, Y., Partyka, A., & Harasymchuk, O. (2025). Research on automated security incident management in public cloud environments. *CEUR Workshop Proceedings*, 4042, 226–249. <https://ceur-ws.org/Vol-4042/>
15. Vakhula, O., Opirskyy, I., Vorobets, P., Bobko, O., & Kulinich, O. (2025). Research on policy-as-code for implementation of role-based and attribute-based access control. *CEUR Workshop Proceedings*, 3991, 139–157. <https://ceur-ws.org/Vol-3991/>
16. Vakhula, O., & Opirskyy, I. (2024). Research on security-as-code approach for cloud-native applications based on Kubernetes clusters. *CEUR Workshop Proceedings*, 3800, 58–69. <https://ceur-ws.org/Vol-3800/>
17. Sysdig. (2024). 2024 cloud-native security and usage report. <https://www.sysdig.com/2024-cloud-native-security-and-usage-report>
18. Sysdig. (2026). 2026 cloud-native security and usage report. <https://www.sysdig.com/press-releases/2026-usage-report>
19. Amazon Web Services. (n.d.). What is EC2 Image Builder? AWS Documentation. <https://docs.aws.amazon.com/imagebuilder/latest/userguide/what-is-image-builder.html>
20. Microsoft. (n.d.). Azure VM Image Builder overview. Microsoft Learn. <https://learn.microsoft.com/en-us/azure/virtual-machines/image-builder-overview>
21. Google Cloud. (n.d.). Setting up trusted image policies. Google Cloud Documentation. <https://cloud.google.com/compute/docs/images/restricting-image-access>
22. Broadcom. (n.d.). Manage VM templates in a content library. VMware vSphere 8.0 Documentation. <https://techdocs.broadcom.com/us/en/vmware-cis/vsphere/vsphere-8-0/vsphere-virtual-machine-administration/using-content-librariesvsphere-vm-admin/manage-vm-templates-in-a-content-libraryvsphere-vm-admin.html>
23. Harbor. (2025). Vulnerability scanning (Harbor 2.13.0 documentation). <https://goharbor.io/docs/2.13.0/administration/vulnerability-scanning/>
24. National Institute of Standards and Technology. (2022). Secure Software Development Framework (SSDF), version 1.1 (NIST SP 800-218). <https://doi.org/10.6028/NIST.SP.800-218>
25. Supply-chain Levels for Software Artifacts. (2026). SLSA specification, version 1.2. <https://slsa.dev/spec/v1.2/>
26. Verizon. (2026). 2026 data breach investigations report. Verizon Business. <https://www.verizon.com/business/resources/reports/dbir/>
27. National Institute of Standards and Technology. (2012). Guide for conducting risk assessments (NIST SP 800-30 Rev. 1). <https://doi.org/10.6028/NIST.SP.800-30r1>
28. FIRST. (2023). Common Vulnerability Scoring System version 4.0: Specification document. <https://www.first.org/cvss/v4.0/specification-document>
29. FIRST. (n.d.). Exploit Prediction Scoring System (EPSS): Model and documentation. <https://www.first.org/epss/>
30. Cybersecurity and Infrastructure Security Agency. (n.d.). Stakeholder-Specific Vulnerability Categorization (SSVC). <https://www.cisa.gov/resources-tools/resources/stakeholder-specific-vulnerability-categorization-ssvc>
31. Cybersecurity and Infrastructure Security Agency. (n.d.). Known Exploited Vulnerabilities catalog.



- <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>
32. Canonical. (n.d.). Ubuntu cloud images. <https://cloud-images.ubuntu.com/>
33. Canonical. (n.d.). Ubuntu Snapshot Service. <https://snapshot.ubuntu.com/>; <https://ubuntu.com/server/docs/how-to/software/snapshot-service/>
34. Aqua Security. (n.d.). Trivy documentation. <https://trivy.dev/docs/>
35. HashiCorp. (n.d.). QEMU builder for Packer. HashiCorp Developer. <https://developer.hashicorp.com/packer/integrations/hashicorp/qemu/latest/components/builder/qemu>
36. War, A., Diallo, A., Habib, A., Klein, J., & Bissyandé, T. F. (2025). Vulnerabilities in infrastructure as code: What, how many, and who? *Empirical Software Engineering*, 30(5), Article 120. <https://doi.org/10.1007/s10664-025-10672-8>
37. Kumar, R., & Goyal, R. (2020). Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). *Computers & Security*, 97, Article 101967. <https://doi.org/10.1016/j.cose.2020.101967>
38. Rajapakse, R. N., Zahedi, M., Babar, M. A., & Shen, H. (2022). Challenges and solutions when adopting DevSecOps: A systematic review. *Information and Software Technology*, 141, Article 106700. <https://doi.org/10.1016/j.infsof.2021.106700>
39. Paul, A., Manoj, R., & Udhayakumar, S. (2024). Amazon Web Services cloud compliance automation with Open Policy Agent. In *Proceedings of the 2024 International Conference on Expert Clouds and Applications (ICOECA)* (pp. 313–317). <https://doi.org/10.1109/ICOECA62351.2024.00063>
40. Diaz-De-Arcaya, J., Osaba, E., Benguria, G., Etxaniz, I., Lobo, J. L., Alonso, J., Torre-Bastida, A. I., & Almeida, A. (2023). IEM: A unified lifecycle orchestrator for multilingual IaC deployments. In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering* (pp. 195–199). <https://doi.org/10.1145/3578245.3584938>
41. Morales, J. A., Hamed, J., Reynolds, D., Shepard, D., Antunes, L., Yankel, J., & Yasar, H. (2023). Experiences with secure pipelines in highly regulated environments. In *Proceedings of the 18th International Conference on Availability, Reliability and Security (ARES 2023)*, Article 57. <https://doi.org/10.1145/3600160.3605466>

Отримано редакцією журналу / Received: 02.09.26

Прорецензовано / Revised: 16.09.26

Схвалено до друку / Accepted: 26.10.26



This work is licensed under Creative Commons Attribution-noncommercial-sharealike 4.0 International License.