УДК 004.8:004.056
DOI: 10.31673/2409-7292.2025.024887

Лещенко Б. С.

КОМПЛЕКСНА МОДЕЛЬ ІНТЕГРАЦІЇ БЕЗПЕКИ У ЖИТТЄВИЙ ЦИКЛ РОЗРОБКИ ДЛЯ ХМАРНИХ СЕРЕДОВИЩ

У цьому дослідженні пропонується комплексна модель, спеціально розроблена для вирішення проблем безпеки, пов'язаних із сучасними хмарними інфраструктурами. Запропонована модель забезпечує впровадження заходів безпеки від початкового планування до завершення життєвого циклу застосунку, надаючи пріоритет безперервному впровадженню безпеки на всіх етапах. Модель орієнтована на інтеграцію безпеки як невід'ємної складової розробницького процесу. Вона передбачає постійне управління ризиками, регулярні аудити та стимулювання безперервних інновацій у межах SDLC. Серед інших ключових компонентів розширеної моделі – управління безпекою (security governance), безпечне виведення з експлуатації компонентів, моніторинг, реагування, навчання та масштабування.

Розширена модель охоплює 20 ключових компонентів, що формують повний набір дій, необхідних для безпечної розробки, впровадження та супроводу сучасних програмних систем. Вона враховує не лише технічні аспекти, але й культурні та процедурні чинники, що є основою для сталого управління безпекою.

Порівняння з існуючими моделями демонструє, що розширена модель не лише усуває прогалини в сучасних практиках, а й пропонує масштабоване рішення, яке відповідає динамічній природі сучасних IT-середовищ. Акцент моделі на постійному впровадженні інновацій і адаптації допомагає організаціям залишатися на крок попереду нових загроз і змін у вимогах до безпеки.

Ключові слова: життєвий цикл розробки програмного забезпечення, SDLC, DevSecOps, безпека хмарних середовищ, управління безпекою, безперервна інтеграція.

Вступ

У сучасному динамічному IT-середовищі інтеграція заходів безпеки безпосередньо в процеси розробки та експлуатації програмного забезпечення стає не просто бажаною, а необхідною. Саме цю потребу задовольняє підхід DevSecOps – еволюційне продовження моделі DevOps, у якому безпека вбудовується у всі етапи життєвого циклу розробки програмного забезпечення (Software Development Lifecycle або SDLC). Основна ідея полягає в тому, щоб ідентифікувати та усувати потенційні вразливості ще на ранніх етапах розробки, а не відкладати розв’язання питань безпеки до завершальних етапів проєкту.

Перехід до DevSecOps зумовлений тим, що традиційні підходи до забезпечення безпеки часто не встигають за темпами впровадження сучасних IT-рішень. DevSecOps вирішує цю проблему шляхом інтеграції механізмів безпеки в CI/CD-процеси (Continuous Integration and Continuous Deployment), що дає змогу виявляти вразливості та реагувати на них ще до того, як вони стануть реальною загрозою. Такий підхід не лише підвищує загальний рівень захищеності систем, а й дозволяє ефективно збалансувати швидкість розробки з вимогами до кібербезпеки.

Зі зростанням кількості організацій, що переходять до хмарних середовищ, потреба в інтегрованих засобах захисту стає все більш актуальною. Хоча контейнери значно спрощують масштабування та розгортання застосунків, вони також можуть відкривати нові вектори для зловмисних атак у разі недостатнього захисту. Отже, забезпечення безпеки хмарних середовищ є критично важливим для захисту застосунків і даних від потенційних інцидентів [1].

Інтеграція традиційних засобів забезпечення безпеки в середовище DevOps супроводжується низкою проблем. Серед основних викликів – невідповідність між статичними підходами до безпеки та динамічними циклами розробки, складність автоматизації процесів захисту, труднощі інтеграції з існуючими інструментами, проблеми з управлінням конфігураціями, вразливості контейнерів, а також організаційні й культурні бар’єри. Подолання цих перешкод є ключовим чинником для успішного впровадження DevSecOps – підходу, що дає змогу створювати безпечне програмне забезпечення швидко та ефективно [2].

Застосування практик DevSecOps має особливе значення для захисту хмарних середовищ, оскільки дозволяє вирішувати специфічні виклики кібербезпеки, властиві цим технологіям. Завдяки цьому підходу формується більш надійна й стійка IT-інфраструктура, що істотно підвищує загальний рівень кіберзахисту в умовах сучасної розробки програмного забезпечення.

Мета та завдання дослідження

Метою цього дослідження є формалізація практик у межах DevSecOps-підходу та їх інтеграція у загальний процес розробки програмного забезпечення з особливим акцентом на безпеку в хмарних середовищах. Запропонований підхід передбачає впровадження механізмів безпеки на всіх етапах розробки та експлуатації з урахуванням принципів DevSecOps – для проактивного виявлення та усунення вразливостей.

Рання ідентифікація потенційних загроз дозволяє значно знизити ризик витоків даних та інших інцидентів інформаційної безпеки. У підсумку дослідження спрямоване на формування кращих практик впровадження DevSecOps у хмарних середовищах. Посилення загальної стійкості та підвищення рівня захищеності IT-інфраструктури розглядається як один із ключових очікуваних результатів.

Дослідження має кілька взаємопов'язаних цілей. Насамперед, передбачається ідентифікація та аналіз актуальних викликів у сфері безпеки програмного забезпечення, зокрема обмежень існуючих методів забезпечення безпеки. У межах дослідження буде також розроблено підходи до безшовної інтеграції засобів безпеки в процес розробки, що дозволить створити адаптивну модель.

Кінцева мета полягає в тому, щоб надати організаціям практичні рекомендації щодо ефективного впровадження DevSecOps у хмарних середовищах, з акцентом на забезпечення безпеки як пріоритету з найперших етапів життєвого циклу розробки.

Окрему увагу приділено розширенню традиційної DevSecOps-моделі шляхом включення додаткових етапів та практик, таких як управління безпекою (security governance), безперервні інновації (continuous innovation) та безпечне виведення з експлуатації компонентів (secure decommissioning). Такий підхід дозволяє комплексно охопити як технологічні, так і організаційні аспекти безпеки в сучасних IT-інфраструктурах.

Аналіз останніх досліджень і публікацій

Розуміння життєвого циклу розробки програмного забезпечення є ключовим, оскільки він забезпечує структурований підхід до управління розробкою, сприяючи ефективності, безпеці та раціональному використанню ресурсів на кожному етапі проєкту. SDLC надає чітку методику, яка охоплює всі етапи – від проєктування та планування до впровадження й супроводу – із належною увагою до критично важливих аспектів на кожному кроці.

Інтеграція вимог безпеки в усі етапи SDLC дає можливість суттєво підвищити захищеність хмарних застосунків. Такий підхід стає особливо важливим для дослідників і практиків, які прагнуть створювати стійкі до загроз рішення в умовах динамічного хмарного середовища.

Ряд досліджень [3, 4] проаналізували етапи та моделі, притаманні різним підходам до SDLC. Процес розробки, тестування та впровадження програмного забезпечення загалом визначається як життєвий цикл розробки програмного забезпечення. Цей цикл включає кілька ключових етапів: планування, проєктування, реалізацію, тестування, розгортання та супровід. На кожному з етапів виконуються специфічні завдання, зокрема збір вимог, програмування, виявлення й усунення помилок, а також розгортання програмного продукту.

Існує кілька підходів до організації етапів розробки програмного забезпечення – зокрема Waterfall, Iterative, Spiral та Agile. Ці моделі надають структуровану основу, яка сприяє ефективному плануванню, підвищенню прозорості процесів, управлінню ризиками та задоволенню потреб замовника, що загалом спрощує керування SDLC [5].

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

Розуміння сильних і слабких сторін кожного з підходів дозволяє командам приймати обґрунтовані рішення і обирати найбільш стійку та придатну до конкретного контексту стратегію, що у підсумку підвищує шанси на успішну реалізацію проєкту [6].

Як зазначають Olorunshola та ін., компаніям часто складно визначитися з єдиною моделлю SDLC, тому доцільно комбінувати декілька підходів у межах одного проєкту [7].

Втім, навіть попри різноманіття доступних методик, питання інтеграції безпеки в SDLC усе ще залишається складним завданням – більшість моделей не охоплює його повною мірою.

У своєму дослідженні Rafiq Ahmad та ін. [8] провели систематичний огляд літератури та класифікували ключові наукові роботи у сфері DevSecOps. Вони ідентифікували 145 типових ризиків безпеки та 424 найкращі практики управління ними. На основі аналізу було запропоновано шість основних етапів DevSecOps-процесу: розробка вимог (requirement engineering), проєктування, розробка/програмування, тестування, розгортання та супровід.

Така поетапна структура дає змогу як дослідникам, так і практикам формувати ефективні стратегії безпеки, здатні враховувати складнощі, пов’язані з хмарними середовищами, і сприяти створенню більш стійких та захищених застосунків.

Термін DevSecOps описує культуру організаційної взаємодії між розробкою (Dev), безпекою (Sec) та експлуатацією (Ops). Основна мета DevSecOps – забезпечити швидке та безпечне постачання програмного коду [9].

Традиційно, безпека розглядається як окремий етап, що виконується після завершення основного циклу розробки, а іноді навіть уже після розгортання програмного забезпечення. Проте з появою підходу DevSecOps ситуація кардинально змінилася: заходи безпеки інтегруються на всіх етапах життєвого циклу розробки, формуючи цілісну та проактивну стратегію захисту.

Одним із ключових принципів сучасного підходу є безперервна безпека (Continuous Security), яка передбачає постійний моніторинг та оперативне виявлення вразливостей на кожному етапі SDLC-циклу. Такий підхід забезпечує не лише швидке реагування на потенційні загрози, а й дозволяє зменшити витрати на усунення вразливостей, виявляючи їх на ранніх стадіях [9].

Основна ідея DevSecOps полягає у принципі “зміщення вліво” (shift left) [2], тобто інтеграції заходів безпеки на ранніх етапах розробки. Такий підхід дозволяє виявляти та усувати вразливості ще до того, як вони потраплять у пізніші етапи життєвого циклу, що суттєво знижує витрати та складність їх усунення.

У своїй роботі Kumar і R. Goyal [10] запропонували поетапну модель реалізації безперервної безпеки, яка охоплює 12 кроків. Ця модель доповнює традиційні етапи життєвого циклу розробки програмного забезпечення (SDLC) і пропонує більш деталізований погляд на інтеграцію безпеки в процес розробки.

DevSecOps представляє собою суттєвий етап еволюції IT-операцій, поєднуючи швидкість і гнучкість DevOps із надійними механізмами забезпечення безпеки. Роль цієї методології у хмарних архітектурах особливо важлива, адже DevSecOps не лише підвищує рівень захищеності систем, а й сприяє кращій співпраці між командами розробки та експлуатації. У міру того як організації все активніше впроваджують хмарні технології та переходять до складних архітектур на основі мікросервісів, DevSecOps стає ключовим елементом для забезпечення безпечної, надійної та ефективної роботи як людей, так і IT систем.

У науковій літературі представлено низку підходів до реалізації безперервної безпеки, однак одним із найґрунтовніших є дослідження Xiaofan Zhao та ін. [11]. Запропонована ними модель *Challenge-Practice-Tool-Metric (CPTM)* є цілісним фреймворком для ефективної інтеграції безпеки в процеси DevOps. Цей підхід базується на огляді літератури (Multi-vocal Literature Review, MLR) і демонструє взаємозв’язки між основними викликами, практиками, інструментами та метриками у життєвому циклі DevSecOps.

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

Використовуючи модель CPTM, автори ідентифікували ключові бар'єри для впровадження DevSecOps – зокрема, опір на рівні організації, складнощі інтеграції та постійну потребу в дотриманні вимог відповідності. Ці проблеми класифікуються за чотирма напрямами: організаційними, процедурними, технологічними та бізнесовими, що дозволяє глибше зрозуміти комплексність викликів, які стоять на шляху до успішного впровадження DevSecOps.

У цьому контексті *модель ADOC* [10] також пропонує цілісний фреймворк, що об'єднує процеси розробки, захисту й експлуатації, гарантуючи інтегрованість практик безпеки на кожному етапі DevOps-пайплайнів. Ключову роль у цьому відіграють програмне забезпечення з відкритим вихідним кодом (Open Source Software, OSS) та хмарні технології, які виступають рушіями моделі. Вона включає шість вимірів, дев'ять рекомендацій, дванадцятиетапний процес та сім практичних напрямів.

Модель ADOC слугує дорожньою картою для організацій, що прагнуть впроваджувати принципи DevSecOps з використанням відкритого ПЗ у хмарних середовищах. Це дозволяє закласти безпеку на кожному етапі розробки, автоматизувавши процеси, що робить створення надійного та якісного програмного забезпечення більш доступним і економічно обґрунтованим. Дванадцятиетапний підхід моделі сприяє тому, щоб безпека розглядалася не як зовнішнє доповнення, а як невід'ємна складова життєвого циклу DevOps, підвищуючи загальний рівень кіберстійкості організації.

У контексті категоризації інструментарію DevSecOps, GitHub-репозиторій від sottlmarek [12] виступає як авторитетне та широко визнане джерело, що підтверджується понад 6000 зірками від спільноти. Його цінність полягає у ретельному підборі та систематизації рішень для впровадження безпеки в DevOps-процеси, з особливим фокусом на хмарні технології та сучасні підходи до DevSecOps. Автори пропонують чітку структуру, групуючи інструменти за такими доменами: перевірки на етапі pre-commit, управління конфігураційними даними та секретами, контроль OSS та залежностей, безпека ланцюга постачання ПЗ, статичний (SAST) і динамічний (DAST) аналіз коду, забезпечення безпеки на етапі безперервного розгортання, а також, окремо, безпека інфраструктури Kubernetes та контейнерів.

Фундація OWASP® робить вагомий внесок у безпеку програмного забезпечення, підтримуючи проєкти, керовані спільнотою, розвиваючи глобальну мережу осередків та організовуючи конференції. Сучасна індустрія розробки програмного забезпечення значною мірою визначається гнучкими методологіями (Agile) та практиками DevOps. У відповідь на виклики, OWASP® активно популяризує проєкти з відкритим кодом, поширює знання та практики безпечної розробки, а також організовує тематичні конференції для обміну досвідом. Важливим інструментом, розробленим Фундацією для системного підходу до планування безпеки, є *модель DevSecOps Maturity* [13].

Модель DevSecOps Maturity наочно демонструє механізми контролю безпеки, які застосовуються при реалізації практик DevOps, а також методики їхнього порівняльного оцінювання (бенчмаркінгу). Важливо зазначити, що практики DevOps самі по собі сприяють підвищенню рівня безпеки, наприклад, через ретельний аналіз кожного компонента образу Docker – від бібліотек застосунків до операційної системи – на предмет відомих вразливостей. Прогресивний підхід, закладений у модель DevSecOps Maturity, слугує орієнтиром для розробки та впровадження актуальних ідей та практичних заходів, спрямованих на ефективне протистояння цим загрозам.

Отже, проаналізовані дані свідчать, що «білі» та «сірі» дослідження не дотримуються єдиної, сталої класифікації життєвого циклу розробки програмного забезпечення у контексті забезпечення безпеки хмарних середовищ. Такий стан речей вказує на активний пошук та застосування різноманітних підходів та наукових орієнтирів у цій дослідницькій царині, що, у свою чергу, сприяє розробці більш широкої та всеохопної моделі. Більше того, це підкреслює

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

об'єктивну складність та багатоаспектність процесу інтеграції принципів DevSecOps у життєвий циклк розробки програмного забезпечення.

Основна частина

Цей розділ присвячено детальному порівняльному аналізу чотирьох провідних моделей: Моделі безперервного планування та тестування (Continuous Planning and Testing Model, CPTM), моделі DevSecOps Maturity, моделі контролю доставки та експлуатації застосунків (Application Delivery and Operations Control model, ADOC) та узагальненого підходу на основі Ultimate DevSecOps library.

Continuous Planning and Testing Model (CPTM) пропонує динамічний, циклічний підхід до забезпечення безпеки в межах життєвого циклу розробки програмного забезпечення (SDLC). Усе починається з етапу планування, на якому визначаються вимоги до безпеки. Далі йде етап створення, де акцент зроблено на проєктуванні захищених архітектур і рішень. На етапі реалізації модель передбачає безперервну перевірку, що дозволяє виявляти помилки безпеки ще до релізу. На етапі тестування головна увага приділяється активному пошуку вразливостей, щоб мінімізувати ризики перед впровадженням. Нарешті, етап релізу охоплює практики безпечного розгортання.

Особливість CPTM полягає в тому, що після випуску програмного забезпечення модель переходить до етапу підтримки, який реалізується як безперервний цикл дій: запобігання, виявлення, реагування, прогнозування та адаптація. Такий підхід підкреслює важливість постійного моніторингу, гнучкості та здатності реагувати на нові виклики для підтримання належного рівня безпеки в умовах динамічних середовищ.

Модель DevSecOps Maturity описує поетапний та системний підхід до масштабування практик безпеки в межах життєвого циклу розробки програмного забезпечення. На етапі планування безпека враховується ще на стадії збору вимог, де безпека стає невіддільною частиною загального бачення проєкту. Далі, на етапі проєктування, безпекові механізми інтегруються безпосередньо в архітектуру системи – безпека перестає бути «доповненням» і стає основою дизайну.

У процесі реалізації модель акцентує увагу на безпечній розробці: використання принципів безпечного написання коду та проведення регулярних оцінок безпеки стають звичною практикою. Етап тестування передбачає, що безпекові перевірки є повноцінною складовою загального забезпечення якості. На етапі впровадження DevSecOps підхід дозволяє інтегрувати перевірки безпеки безпосередньо в CI/CD-пайплайни, роблячи ці перевірки частиною релізного циклу. Нарешті, під час етапу підтримки, модель гарантує, що безпека зберігає пріоритетний статус протягом усього життєвого циклу продукту, включаючи моніторинг, оновлення та реагування на інциденти.

Application Delivery and Operations Control (ADOC) пропонує комплексний підхід, який поєднує управління як доставкою програмного забезпечення, так і його експлуатацією. Модель починається з етапу планування, де безпека інтегрується у стратегічне бачення розвитку продукту. Далі, під час етапу розробки коду, особливу увагу приділено практикам безпечного написання коду – вони розглядаються як ключовий елемент архітектурного проєктування.

На етапі реалізації акцент робиться на перевірених і безпечних змінах у коді: кожна зміна коду має відповідати встановленим стандартам надійності та безпеки. Під час тестування та впровадження активності збирання і інтеграції впроваджують контроль безпеки безпосередньо в CI/CD-процеси, що забезпечує безперервний моніторинг і перевірку.

Етап підтримки у моделі ADOC вирізняється високою деталізацією: вона охоплює такі дії, як пакування, реліз, конфігурування, приймання, розгортання, експлуатація та адаптація. Ці дії працюють у тісному зв’язку, щоб гарантувати, що безпека не лише підтримується, а й еволюціонує відповідно до нових викликів, змін у середовищі та загроз.

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

Ultimate DevSecOps Library пропонує узагальнений, але цілісний підхід до впровадження практик DevSecOps, орієнтуючись на класичні етапи життєвого циклу розробки програмного забезпечення (SDLC), із системним включенням заходів безпеки на кожному з них. На етапі планування відбувається виявлення та інтеграція вимог безпеки, що закладає основу для подальшого розвитку безпечного продукту. Етап розробки охоплює використання перевірених практик безпечного написання коду, а збірка зосереджене на впровадженні контролів безпеки безпосередньо в процес побудови артефактів.

На етапі тестування пріоритет надається глибокій перевірці безпеки – від статичного й динамічного аналізу до інтеграційних тестів, орієнтованих на виявлення вразливостей до моменту впровадження. Етап релізу забезпечує безпечне розгортання програмного забезпечення з урахуванням політик контролю доступу, інфраструктурних конфігурацій та обмеження привілеїв.

Останній етап – підтримка – включає в себе активності розгортання, експлуатації та моніторингу, які гарантують постійний контроль і реакцію на інциденти. Такий підхід забезпечує цілісність безпеки не лише під час розробки, а й упродовж усього життєвого циклу програмного забезпечення в хмарному середовищі.

Таблиця 1

Порівняння моделей

CPTM [11]	**DevSecOps Maturity [13]**	**ADOC [10]**	**DevSecOps library [12]**
Plan	Requirements Gathering	Plan	Plan
Create	Design	Code	Code
Verify	Development	Commit	Build
Preprod	Testing	Build	Test
Release	Deployment	Integrate	Release
Prevent	Maintenance	Package	Deploy
Detect		Release	Operate
Respond		Configure	Monitor
Predict		Accept	
Adapt		Deploy	
		Operate	
		Adapt	

Порівняльний аналіз (табл. 1) демонструє, що кожна з розглянутих моделей має свої унікальні сильні сторони та зосереджена на різних аспектах, однак усі вони переслідують спільну мету – інтеграцію безпеки на всіх етапах життєвого циклу розробки програмного забезпечення (SDLC). Таке порівняння дозволяє отримати глибші уявлення про те, як безпекові механізми можуть бути системно впроваджені в процес розробки, що, у свою чергу, сприяє зміцненню загальної кіберстійкості сучасних програмних систем.

З огляду на динамічний характер загроз, які постійно «еволюціонують», впровадження, адаптація та вдосконалення цих моделей є критично важливими для організацій, які прагнуть підтримувати високий рівень безпеки на всіх етапах життєвого циклу програмного забезпечення. Такі моделі слугують основою для формування стратегічно обґрунтованих, гнучких і масштабованих підходів до безпеки у DevOps-середовищах.

Результати

У дослідженні застосовано комплексний методологічний підхід для створення повноцінної DevSecOps-моделі, орієнтованої на захист хмарних середовищ. Методологія

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

націлена на досягнення поставлених дослідницьких цілей і завдань. Першим кроком стало проведення детального огляду літератури, який заклав підґрунтя для розуміння існуючих підходів до DevSecOps, а також типових викликів у сфері безпеки. Було проаналізовано широкий спектр джерел: наукові статті, матеріали конференцій, аналітичні звіти від представників індустрії, а також "сіру" літературу та звіти сторонніх вендорів. Цей аналіз дозволив не лише окреслити еволюцію підходів до безпеки в сучасній IT-інфраструктурі, але й виявити слабкі місця у поточних моделях, що стало основою для наступних етапів дослідження.

Спираючись на результати літературного огляду, було проведено порівняльний аналіз наявних моделей і фреймворків. Оцінювання охоплювало ключові аспекти, зокрема рівень інтеграції безпеки в життєвий цикл розробки, здатність адаптуватися до хмарних середовищ і масштабованість у складних IT-середовищах. Це дало змогу виявити як сильні сторони, так і обмеження існуючих рішень, що, своєю чергою, допомогло чітко окреслити напрями для вдосконалення. Саме цей етап став вирішальним для формування нової, розширеної моделі, яку ми пропонуємо в рамках цього дослідження.

Третій етап дослідження завершився розробкою нової моделі. Спираючись на результати літературного огляду та порівняльного аналізу, було сформовано розширену модель, яка покликана усунути виявлені недоліки та прогалини в наявних підходах.

Структура нової моделі свідомо прив'язана до життєвого циклу розробки програмного забезпечення, що забезпечує її органічне впровадження в наявні процеси розробки. Такий підхід робить модель не лише теоретично обґрунтованою, але й практично придатною до використання в реальних умовах.

Застосовані методи дозволили створити гнучку та стійку модель, орієнтовану на підвищення рівня безпеки в процесах розробки, з особливим акцентом на хмарні середовища. Запропонована модель може стати ефективним інструментом для організацій, що прагнуть інтегрувати безпеку на всіх етапах розробки програмного забезпечення.

У ході нашого дослідження було виявлено низку аспектів, які потребують вдосконалення в існуючих моделях та їх класифікаціях. Зокрема, доцільно розширити типову структуру моделей, додавши етапи, які враховують специфічні виклики та практики, описані раніше.

Одним із ключових напрямів є посилення ролі безперервної освіти. Усі члени команди повинні регулярно проходити навчання щодо актуальних методів забезпечення безпеки, використання відповідних інструментів і реагування на нові загрози. Це сприяє формуванню культури інформаційної безпеки, в якій кожен учасник процесу усвідомлює свою відповідальність.

Інтеграція заходів із забезпечення безперервності бізнесу та відновлення після інцидентів є ще одним важливим елементом. Розробка, тестування та регулярне оновлення таких планів дозволяє зменшити час простою та підвищити стійкість систем до збоїв.

Не менш важливими є періодичні аудити безпеки. За можливості їх варто автоматизувати й інтегрувати безпосередньо в CI/CD-процеси. Це дозволяє постійно відстежувати ефективність застосованих заходів та забезпечувати відповідність стандартам і вимогам до комплаєнсу.

У міру зростання складності застосунків, особливо в середовищах контейнеризації та хмарної інфраструктури, необхідно масштабувати і стратегії безпеки. Адаптація до нових викликів забезпечує релевантність підходів у динамічному технологічному середовищі.

Нарешті, впровадження структурованих політик і фреймворків управління (governance frameworks) дозволяє впевнено дотримуватися як нормативних вимог, так і внутрішніх політик організації. Такі підходи слугують базою для безпечної розробки, впровадження та експлуатації програмних рішень.

Не менш критичним етапом є безпечне виведення з експлуатації компонентів або застосунків, що досягли завершення свого життєвого циклу. Цей процес передбачає повне

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

видалення даних (правильне безпечне архівування), демонтаж інфраструктури та усунення залишкових вразливостей, які потенційно можуть бути використані зловмисниками.

Також, постійне впровадження інновацій у сфері безпеки – як технологічних, так і процесуальних – дозволяє організаціям залишатися на крок попереду нових загроз. Адаптація до змін шляхом використання сучасних інструментів, методів і практик забезпечує гнучкість і стійкість в хмарних середовищах.

Таблиця 2

Запропонована розширена модель

Етап	Опис
Навчання (Educate)	Безпека починається з людей, тому важливо постійно підвищувати кваліфікацію команди – як у технічних аспектах безпеки, так і в розумінні ризиків на рівні процесів.
Планування (Plan)	На ранніх етапах слід формувати чітку стратегію проєкту: визначати цілі, збирати вимоги та ідентифікувати потенційні загрози безпеці ще до початку розробки.
Управління (Govern)	Необхідно впроваджувати рамкові політики безпеки (security governance) та забезпечувати відповідність нормативним вимогам і стандартам.
Написання коду (Code)	Під час написання коду важливо дотримуватися принципів безпечної розробки, щоб зменшити кількість вразливостей ще на цьому етапі.
Фіксація змін (Commit)	Практики безпечної роботи з системами контролю версій (VCS) дозволяють гарантувати цілісність і надійність змін у коді.
Збірка (Build)	Автоматизовані засоби тестування безпеки мають бути інтегровані в процес збірки, щоб швидко виявляти проблеми ще до запуску застосунку.
Інтеграція (Integrate)	Компоненти мають інтегруватися в систему безпечно, з автоматизованою перевіркою сумісності та відсутності конфліктів.
Упакування (Package)	Пакування застосунку має відбуватись із врахуванням вимог до безпеки – включно з перевіркою залежностей і цифровим підписуванням.
Конфігурування (Configure)	Налаштування повинні бути стандартизовані й безпечні – зокрема шляхом застосування принципів "інфраструктури як коду" (IaC).
Реліз (Release)	Перед випуском у продакшн варто проводити фінальну валідацію безпеки, щоб уникнути критичних помилок та/або вразливостей.
Розгортання (Deploy)	Розгортання застосунку має бути автоматизованим та включати інтегровані перевірки безпеки, що гарантують доставку перевіреного коду.
Експлуатація (Operate)	На етапі роботи системи необхідно забезпечити безперервний моніторинг і підтримку в реальному часі.
Моніторинг (Monitor)	Слід використовувати автоматизовані інструменти для безперервного спостереження за безпековим станом системи та виявлення аномалій.
Реагування (Respond)	Має бути чітко визначено, як команда реагуватиме на інциденти безпеки – зокрема з урахуванням сценаріїв, часу реагування та відповідальних осіб.
Аудит (Audit)	Регулярні перевірки безпеки дозволяють оцінити ефективність заходів та виявити напрями для покращення.
Оцінка прийнятності (Accept)	Завершальний етап перевірки, коли система проходить тестування на відповідність встановленим критеріям безпеки та готовності до розгортання.
Масштабування (Scale)	Із ростом системи зростають і вимоги до безпеки – тому стратегії захисту мають еволюціонувати відповідно до нових викликів.
Адаптація (Adapt)	Безпекові практики потребують регулярного перегляду та оновлення відповідно до змін у загрозах, технологіях та процесах.
Інновації (Innovate)	Інтеграція новітніх технологій і підходів до безпеки (наприклад, zero trust або AI-driven security) є критично важливою для довгострокової стійкості.
Виведення з експлуатації (Decommission)	Завершення життєвого циклу систем або компонентів має відбуватися з дотриманням усіх вимог до безпечної утилізації або архівування.

Запропонована нами розширена модель (табл. 2) охоплює 20 ключових компонентів, що формують повний набір дій, необхідних для безпечної розробки, впровадження та супроводу сучасних програмних систем. Додавання таких етапів, як моніторинг, реагування, аудит і

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

навчання, дозволяє моделі виходити за межі суто технічних аспектів і охоплювати також культурні та процедурні чинники, які є основою для сталого управління безпекою.

Ця модель особливо ефективна у контексті хмарних середовищ, де інтеграція безпеки в усі етапи життєвого циклу ПЗ є необхідною умовою. Її можна успішно застосовувати під час проєктування та автоматизації розгортання корпоративних підсистем безпеки.

Модель охоплює всі етапи життєвого циклу розробки програмного забезпечення (SDLC) і інтегрує вимоги безпеки на кожному з етапів. Таким чином, вона формує системний підхід до створення безпечних програмних рішень у хмарному середовищі нового покоління.

Висновки

У цьому дослідженні запропоновано комплексну модель, спеціально розроблену для вирішення проблем безпеки в хмарних середовищах. Впроваджуючи практики безпеки на всіх етапах життєвого циклу розробки програмного забезпечення, ця модель створює міцну основу для підвищення рівня захищеності організацій. Запропонована модель охоплює ключові компоненти, такі як управління безпекою, регулярні аудити та безпечне виведення з експлуатації, що забезпечує безперервну й інтегровану безпеку в процесі розробки.

Порівняння з існуючими фреймворками демонструє, що розширена модель не лише усуває прогалини в сучасних практиках, а й пропонує масштабоване рішення, яке відповідає динамічній природі сучасних IT-середовищ. Акцент моделі на постійному впровадженні інновацій і адаптації допомагає організаціям залишатися на крок попереду нових загроз і змін у вимогах до безпеки.

Емпірична перевірка ефективності та масштабованості запропонованого підходу DevSecOps через його впровадження в реальних застосунках у різних галузях є важливим напрямом подальших досліджень. Окрему увагу варто приділити інтеграції автоматизованих інструментів безпеки, зокрема у CI/CD-пайплайни, щоб дослідити, як ці інструменти можна ефективно впровадити в запропоновану парадигму.

Подальші дослідження можуть зосередитися на вивченні організаційних та культурних бар'єрів, що заважають впровадженню методів DevSecOps, а також на тому, як навчальні ініціативи та стратегічне управління змінами можуть допомогти подолати ці перешкоди. Важливо також оцінити адаптивність моделі до нових технологічних трендів, таких як безсерверні обчислення (serverless) та процеси розробки, керовані штучним інтелектом.

Ще одним перспективним напрямом є дослідження впливу запропонованої моделі на ефективність і продуктивність, зокрема в контексті розподілу ресурсів команд розробників і скорочення часу виходу продукту на ринок. Нарешті, майбутні роботи мають зосередитися на узгодженні моделі з різними нормативними вимогами та вивченні можливості автоматизації перевірок відповідності (compliance checks) у рамках DevSecOps-процедур.

Опрацювання цих напрямів досліджень дозволить удосконалити та адаптувати запропоновану модель відповідно до змінних потреб індустрії розробки програмного забезпечення, що значно сприятиме створенню безпечної та стійкої IT-інфраструктури.

Перелік посилань

1. DATA BREACH MANAGEMENT: AN INTEGRATED RISK MODEL / F. Khan та ін. Information & Management. 2021. Т. 58, № 1. С. 103392. URL: https: // doi.org / 10.1016 / j.im.2020.103392 (дата звернення: 25.05.2025).

2. Rajapakse R., Zahedi M., Babar M. Challenges and solutions when adopting DevSecOps: A systematic review. Journal of Information and Software Technology. 2021.

3. Ruparelia N. B. Software development lifecycle models. ACM SIGSOFT Software Engineering Notes. 2010. Т. 35, № 3. С. 8–13. URL: https://doi.org/10.1145/1764810.1764814 (дата звернення: 25.05.2025).

4. Jain R., Suman U. A Systematic Literature Review on Global Software Development Life Cycle. ACM SIGSOFT Software Engineering Notes. 2015. Т. 40, № 2. С. 1–14. URL: https://doi.org/10.1145/2735399.2735408 (дата звернення: 25.05.2025).

5. Acharya B., Sahu P. Software Development Life Cycle Models: A Review Paper. International Journal of Advanced Research in Engineering and Technology. 2020. Т. 11. С. 169–176. URL: https://doi.org/ 10.34218/ IJARET.11.12.2020.019.

© Лещенко Б. С. (2025). Комплексна модель інтеграції безпеки у життєвий цикл розробки для хмарних середовищ. Сучасний захист інформації, 2(62), 88–97.
https://doi.org/10.31673/2409-7292.2025.024887

6. Amazon Web Services I. What is SDLC? - Software Development Lifecycle Explained. URL: https: // aws. amazon. com / what-is / sdlc / #:~:text=The % 20software %2 0development % 20lifecycle%20 (SDLC, expectations% 20during%20production%20and%20beyond(дата звернення: 25.05.2025).

7. Olorunshola O., Ogwueleka F. Review of System Development Life Cycle (SDLC) Models for Effective Application Delivery. Lecture Notes in Networks and Systems. 2021.

8. Systematic Literature Review on Security Risks and its Practices in Secure Software Development / R. A. Khan та ін. IEEE Access. 2022. Т. 10. С. 5456–5481. URL: https://doi.org/10.1109/access.2022.3140181(дата звернення: 25.05.2025).

9. Solutions - DevSecOps - Addressing Security Challenges in a Fast-Evolving Landscape White Paper. Cisco. URL: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/devsecops-addressing-security-challenges.html(дата звернення: 25.05.2025).

10. Kumar R., Goyal R. Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC). Computers & Security. 2020. Т. 97. С. 101967. URL: https://doi.org/10.1016/j.cose.2020.101967(дата звернення: 25.05.2025).

11. Zhao X., Clear T., Lal R. Identifying the primary dimensions of DevSecOps: A multi-vocal literature review. Journal of Systems and Software. 2024. С. 112063. URL: https://doi.org/10.1016/j.jss.2024.112063(дата звернення: 25.05.2025).

12. GitHub - sottlmarek/DevSecOps: Ultimate DevSecOps library. GitHub. URL: https://github.com/sottlmarek/DevSecOps(дата звернення: 25.05.2025).

13. OWASP Devsecops Maturity Model | OWASP Foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/www-project-devsecops-maturity-model/(дата звернення: 25.05.2025).

Надійшла 09.05.2025