КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (32), 2026
ISSN 2663 – 4023
DOI 10.28925/2663-4023.2026.32.1184
УДК 004.415.2.056

Гнатюк Сергій Олександрович
доктор технічних наук, професор, проректор з наукових досліджень та трансферу технологій
Державний університет «Київський авіаційний інститут», Київ, Україна
ORCID: 0000-0003-4992-0564
s.gnatyuk@kai.edu.ua

Побережна Заріна Миколаївна
доктор економічних наук, професор, професор кафедри національної безпеки та підприємництва
Державний університет «Київський авіаційний інститут», Київ, Україна
ORCID: 0000-0001-6245-038X
zarina.poberezhna@npp.kai.edu.ua

Скуратівський Анатолій Анатолійович
аспірант Phd, кафедри комп'ютерних інформаційних технологій
Державний університет «Київський авіаційний інститут», Київ, Україна
ORCID: 0009-0000-0566-2394
a.skurativskyi@gmail.com

МЕТОД ІНТЕГРУВАННЯ ВИМОГ КІБЕРБЕЗПЕКИ В ЖИТТЄВИЙ ЦИКЛ РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Анотація. У сучасних умовах важливим є не лише розроблення програмного забезпечення для захисту інформаційних систем та даних, а інтегрування функцій безпеки (кібербезпеки) до фаз життєвого циклу розроблення програмного забезпечення. З огляду на це, в роботі було формалізовано у загальному вигляді відомі моделі життєвого циклу розроблення програмного забезпечення і сформовано уніфіковану модель SDLC. Також, було розроблено метод інтегрування вимог кібербезпеки в SDLC, що дозволяє інтегрувати вимоги кібербезпеки в конкретну фазу (pipeline) життєвого циклу розроблення програмного забезпечення відповідно до моделі DevSecOps, також дає можливість формалізованої оптимізації вибору контролів кібербезпеки залежно від контексту системи та ресурсних обмежень. Отримані результати можуть бути використані для системного інтегрування вимог кібербезпеки (згідно нормативних документів ISO/IEC, NIST, PCI DSS, PSD2, GDPR, MITRE ATT&CK) в процеси розроблення програмного забезпечення в організаціях, що створюють або експлуатують інформаційні системи критичної інфраструктури, хмарні сервіси та корпоративні інформаційно-комунікаційні системи.

Ключові слова: кібербезпека; життєвий цикл розроблення ПЗ; SDLC; DevSecOps; моделювання загроз; безпека; вимоги кібербезпеки.

ВСТУП

Стрімка цифровізація всіх сфер суспільного життя призвела до того, що програмне забезпечення (ПЗ) стало фундаментом критичної інфраструктури, фінансових систем та державного управління. Разом із цим зростає кількість і складність кібератак, спрямованих на експлуатацію вразливостей у програмних продуктах. Традиційні підходи виявляються менш ефективними для сучасних умов, адже перевірка безпеки відбувається лише на фінальних етапах розроблення, що потребує значних часових та фінансових витрат. Найбільш актуальним вирішенням цієї проблеми є формування концепції, яка передбачає інтеграцію вимог кібербезпеки безпосередньо в життєвий цикл розроблення програмного забезпечення (SDLC).

Впровадження спеціалізованих методів безпечного розроблення дозволяє виявляти загрози ще на етапі проєктування, що забезпечує стійкість системи в цілому.

Сучасний процес розроблення ПЗ характеризується високою швидкістю випуску оновлень та складністю архітектурних рішень. Однак, незважаючи на технологічний прогрес, питання кібербезпеки часто залишаються на етапі розробки. Таким чином, постає нагальна потреба у розробці методу, який дозволив би безперервно та автоматизовано інтегрувати вимоги кібербезпеки в кожен етап життєвого циклу ПЗ, не знижуючи при цьому темпів розробки та забезпечуючи високу стійкість продукту до сучасних кіберзагроз.

Дослідження питань кібербезпеки в процесах програмного забезпечення досліджувалось в багатьох працях різних авторів. В статті [1] автори досліджували математичне моделювання процесів розроблення та підвищення продуктивності SDLC за допомогою штучного інтелекту в контексті безпеки. Проте потребує розроблення методики створення ПЗ від моменту формування ідеї до його завершення. Одним із важливих питань є впровадження стандартів аудиту кібербезпеки та оцінки ризиків у інформаційних системах, що досліджувалось авторами в праці [2]. Варто зазначити, що для забезпечення управління інформаційною безпекою, яка є частиною безпечного життєвого циклу за допомогою методів машинного навчання дозволяє забезпечити достатню швидкість виявлення нових загроз, особливо в умовах постійно зростаючої складності кібератак [3]. Розробка методів пріоритезації кіберінцидентів та оцінки критичності в інформаційних системах знайшли своє відображення в працях авторів [4]. Особливої актуальності набувають питання захисту інформації, де розглядаються аспекти кібербезпеки протягом усього життєвого циклу систем, а також дослідження щодо інтеграції технічних оцінок суворості вразливостей (CVSS) у національні системи моніторингу [5]. Не зважаючи на велику кількість публікацій із зазначеної тематики, потребує дослідження саме методів інтегрування вимог кібербезпеки в диттєвий цикл розроблення ПЗ.

Метою статті є забезпечення системного інтегрування вимог кібербезпеки в життєвий цикл розроблення ПЗ за рахунок розробки й дослідження багатоетапного методу інтегрування вимог з нормативних документів (стандартів і рекомендованих практик) у різні фази SDLC.

ТЕОРЕТИЧНІ ОСНОВИ ДОСЛІДЖЕННЯ

Життєвий цикл розроблення ПЗ (Software Development Life Cycle, SDLC) – це структурований процес створення, впровадження та супроводу ПЗ від моменту формування ідеї до завершення експлуатації [6]. Головною метою SDLC є забезпечення передбачуваності результатів, контролю якості, управління ризиками та оптимального використання ресурсів.

Класично SDLC включає такі фази (рис.1) [7]:

1) збирання вимог та їх аналізування;
2) проєктування дизайну системи;
3) реалізація;
4) тестування;
5) впровадження (розгортання);
6) експлуатація та супровід.

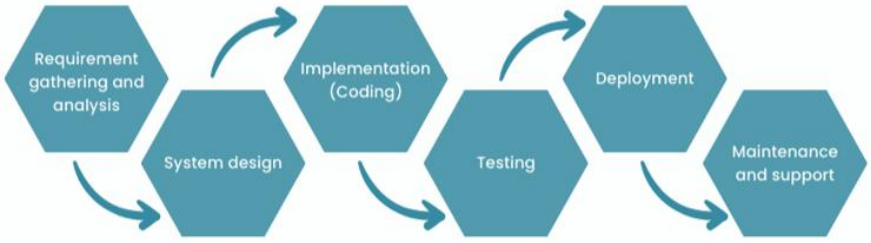


Рис.1. Ключові фази SDLC [7]

У сучасних умовах до цих етапів дедалі частіше інтегруються практики DevOps [8], безперервної інтеграції та безперервної доставки (CI/CD), а також сучасної концепції DevSecOps [9], що передбачає вбудовування механізмів кібербезпеки на всіх фазах життєвого циклу.

У загальному вигляді життєвий цикл розроблення ПЗ можна відобразити за допомогою множини:

$$SDLC = \left\{\bigcup_{i=1}^{n} SDLC_i\right\} = \{SDLC_1, SDLC_2, \dots, SDLC_n\}, \quad (1)$$

де $SDLC_i \subseteq SDLC (i = \overline{1,n})$ – фази життєвого циклу відповідно до певної моделі.

Для прикладу при n=6 (кейс відображено на рис. 1) уніфікована модель (1) матиме наступний вигляд:

$$SDLC_{classic} = \left\{\bigcup_{i=1}^{6} SDLC_i\right\} = \{SDLC_1, SDLC_2, \dots, SDLC_6\} = \{RGA, SYS, IMP, TES, DPL, MTN\}, \quad (2)$$

де $SDLC_1 = RGA, SDLC_2 = SYS, SDLC_3 = IMP, SDLC_4 = TES, SDLC_5 = DPL, SDLC_6 = MTN$ – це відповідно фаза збирання вимог та їх аналізування; фаза проєктування дизайну системи; фаза реалізація; фаза тестування; фаза впровадження (розгортання); фаза експлуатації та супроводу.

Ключовою характеристикою SDLC є наявність формалізованих артефактів на кожному етапі: специфікацій вимог, моделей архітектури, технічної документації, тест-кейсів, звітів верифікації тощо. Від правильності формування вимог значною мірою залежить якість кінцевого продукту, оскільки помилки на ранніх етапах мають кумулятивний ефект і суттєво збільшують вартість їх виправлення на пізніх фазах. Саме тому сучасні підходи роблять акцент на ітеративності, прототипуванні та ранній валідації гіпотез. У випадку систем критичної інфраструктури або оборонного призначення життєвий цикл доповнюється процедурами сертифікації, управління конфігурацією, трасованості вимог та формальної перевірки відповідності стандартам (наприклад, розглянуті в попередніх розділах роботи стандарти: ISO/IEC [10], NIST [11], PCI DSS [12], PSD2 [13], GDPR [14], MITRE ATT&CK [15]).

Існує кілька базових моделей SDLC, які відрізняються ступенем формалізації, гнучкості та способами управління змінами. Розглянемо основні з них:

1) Класична каскадна (водоспадна) модель (Waterfall) (Рис.2) [16] передбачає послідовне проходження етапів, де кожна наступна фаза починається лише після завершення попередньої. Вона добре підходить для проєктів зі стабільними вимогами та високими вимогами до документації, однак є не гнучкою (малогнучкою) у разі змін та впливів.

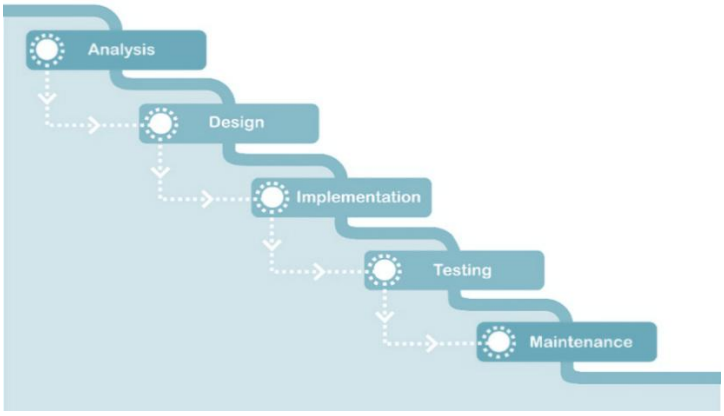


Рис. 2. Waterfall-модель [16]

Відповідно, для формалізації класичної каскадної моделі при *n*=6 (рис. 2) на основі (1) можна представити наступну множину:

$$SDLC_{waterfall} = \left\{\bigcup_{i=1}^{5} SDLC_i\right\} = \{SDLC_1, SDLC_2, \dots, SDLC_6\} = \{RGA, SYS, IMP, TES, MTN\}, \quad (3)$$

де $SDLC_1 = RGA, SDLC_2 = SYS, SDLC_3 = IMP, SDLC_4 = TES, SDLC_5 = MTN$ – це відповідно фаза збирання вимог та їх аналізування; фаза проєктування дизайну системи; фаза реалізації; фаза тестування; фаза експлуатації та супроводу.

2) V-модель (рис.3) [17] є розвитком каскадної, де кожному етапу розроблення відповідає відповідний рівень тестування, що забезпечує чітку кореляцію між верифікацією та валідацією.

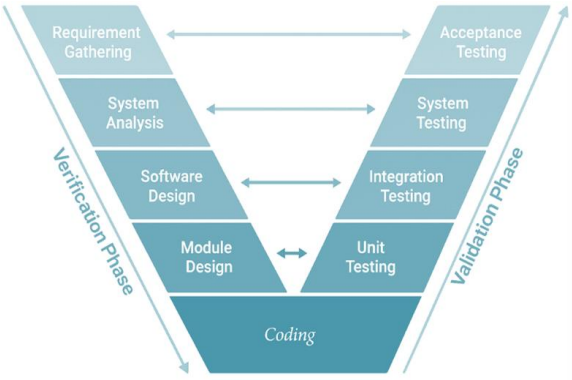


Рис. 3. V-модель [17]

Відповідно, для формалізації V-моделі при *n*=2 (рис.3) на основі (1) можна представити наступну множину:

$$SDLC_V = \left\{\bigcup_{i=1}^{2} SDLC_i\right\} = \{SDLC_1, SDLC_2\} = \{VER, VAL\}, \quad (4)$$

де $SDLC_1 = VER, SDLC_2 = VAL$ – це відповідно фази верифікації та валідації.

Далі, відповідно кожну з підмножин VER та VAL можна представити аналогічним чином.

Для VER при n=5 будемо мати:

$$VER = \left\{\bigcup_{i=1}^{5} VER_i\right\} = \{VER_1, VER_2, \dots, VER_5\} = \{RGA, SYS, SDS, MDS, IMP\}, \quad (5)$$

де $VER_1 = RGA, VER_2 = SYS, VER_3 = SDS, VER_4 = MDS, VER_5 = IMP$ – це фаза збирання вимог та їх аналізування; фаза проєктування дизайну системи; фаза дизайну ПЗ; фаза дизайну модулів; фаза реалізація відповідно.

Для VER при n=5 будемо мати:

$$VAL = \left\{\bigcup_{i=1}^{5} VAL_i\right\} = \{VAL_1, VAL_2, \dots, VAL_5\} = \{IMP, UNT, INT, SYT, ACT\}, \quad (6)$$

де $VAL_1 = IMP, VAL_2 = UNT, VAL_3 = INT, VAL_4 = SYT, VAL_5 = ACT$ – це фаза реалізації, фаза тестування юнітів (компонентів), фаза інтегративного тестування, фаза системного тестування та фаза погодженого тестування відповідно.

З урахуванням (5) та (6) множину (4) і класичну каскадноу V-модель можна представити в узагальненому вигляді наступним чином:

$$\begin{aligned} SDLC_V = \{VER, VAL\} &= \left\{\bigcup_{i=1}^{5} VER_i\right\} \cup \left\{\bigcup_{i=1}^{5} VAL_i\right\} \\ &= \{VER_1, VER_2, \dots, VER_5\} \cup \{VAL_1, VAL_2, \dots, VAL_5\} \\ &= \{RGA, SYS, SDS, MDS, IMP, UNT, INT, SYT, ACT\}, \end{aligned} \quad (7)$$

3) Інкрементальна модель (рис.4) [18] передбачає поетапне створення функціоналу з регулярним зворотним зв'язком. Відповідно до інкрементальної моделі ПЗ розробляється з лінійною послідовністю стадій, але в кілька інкрементів (версій). Таким чином, поліпшення продукту проходить заплановано весь час, поки життєвий цикл розробки ПЗ не завершиться.

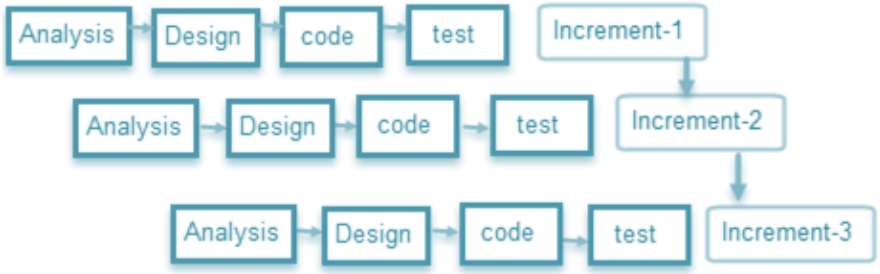


Рис. 4. Інкрементальна модель [18]

У випадку інкрементальної моделі вираз (11) матиме особливості – крім індексу $i = \overline{1,n}$ вводиться індекс $j = \overline{1,m}$ наступним чином:

$$\begin{aligned} SDLC_{increment} &= \left\{\bigcup_{i=1}^{n}\bigcup_{j=1}^{m} SDLC_{ij}\right\} \\ &= \{SDLC_{11}, SDLC_{12}, \dots, SDLC_{nm}\}, (i = \overline{1,n}), (j = \overline{1,m}) \end{aligned} \quad (8)$$

Тоді частковий випадок, відображений на рис. 4, з урахуванням (8) та n=4, m=3 можна представити таким чином:

$$SDLC_{increment} = \left\{\bigcup_{i=1}^{4}\bigcup_{j=1}^{3} SDLC_{ij}\right\} = \{SDLC_{11}, SDLC_{12}, \dots, SDLC_{43}\} = \{RGA_1, SYS_1, IMP_1, TES_1\} \cup \{RGA_2, SYS_2, IMP_2, TES_2\} \cup \{RGA_3, SYS_3, IMP_3, TES_3\} \quad (9)$$

де відповідно за трьома інкрементами (версіями) $\mathrm{RGA}_1, \mathrm{RGA}_2, \mathrm{RGA}_3$ – фази збирання вимог та їх аналізування, $\mathrm{SYS}_1, \mathrm{SYS}_2, \mathrm{SYS}_3$ – фази проєктування дизайну системи, $\mathrm{IMP}_1, \mathrm{IMP}_2, \mathrm{IMP}_3$ – фази реалізації, $\mathrm{TES}_1, \mathrm{TES}_2, \mathrm{TES}_3$ – фази тестування.

4) Спіральна модель (Spiral Model) (рис. 5) [19] орієнтована на управління ризиками та передбачає циклічне проходження етапів із постійною оцінкою загроз і невизначеностей.

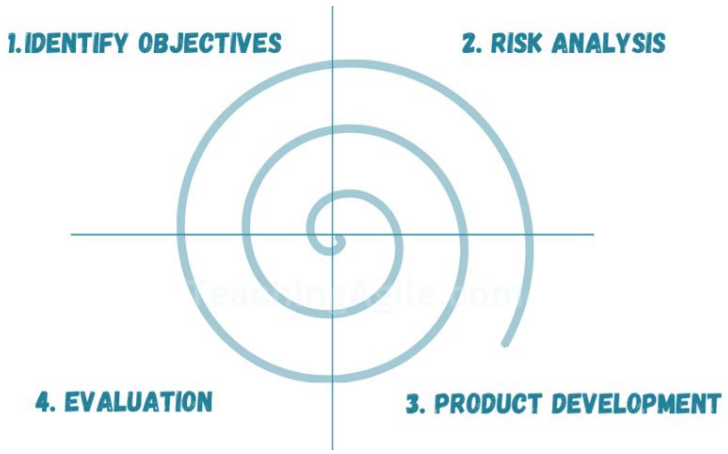


Рис. 5. Спіральна модель [19]

Відповідно, для формалізації спіральної моделі при n=4 (рис.5) на основі (1) можна представити наступну множину:

$$SDLC_{spiral} = \left\{\bigcup_{i=1}^{4} SDLC_i\right\} = \{SDLC_1, SDLC_2, SDLC_3, SDLC_4 \dots\} = \{IDN, RSK, PRD, EVL \dots\}, \quad (10)$$

де $\mathrm{SDLC}_1 = \mathrm{IDN}$ – фаза ідентифікації об’єктів, $\mathrm{SDLC}_2 = \mathrm{RSK}$ – фаза аналізування ризиків, $\mathrm{SDLC}_3 = \mathrm{PRD}$ – фаза розроблення продукту, $\mathrm{SDLC}_4 = \mathrm{EVL}$ – фаза оцінювання, які повторюються циклічно.

5) Гнучкі Agile-підходи (до таких підходів відносяться Scrum, Kanban, Extreme Programming) (рис. 6) [20] акцентують увагу на гнучкості, швидкій адаптації до змін вимог та активній взаємодії із замовником.

Для прикладу, Agile модель при n=5 (рис. 6) із врахуванням базового виразу (1), можна представити у вигляді такої множини:

$$SDLC_{agile} = \left\{\bigcup_{i=1}^{5} SDLC_i\right\} = \{SDLC_1, SDLC_2, SDLC_3, SDLC_4, SDLC_5\} = \{SYS, DEV, TES, DPL, REV\}, \quad (11)$$

де $SDLC_1 = SYS$ – фаза проєктування дизайну системи, $SDLC_2 = DEV$ – фаза розроблення, $SDLC_3 = TES$ – фаза тестування, $SDLC_4 = DPL$ – фаза розгортання й експлуатації, $SDLC_5 = REV$ – фаза експертизи (зворотного зв'язку).

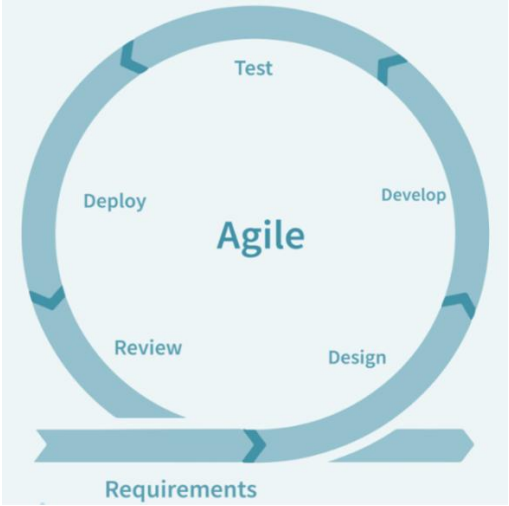


Рис. 6. Agile модель [20]

У рамках Agile розроблення здійснюється короткими ітераціями (sprints), а продукт формується інкрементально. Сучасним розвитком є DevOps-модель (рис. 7), яка інтегрує процеси розроблення та експлуатації, автоматизує тестування, розгортання та моніторинг, забезпечуючи безперервну доставку цінності користувачам.

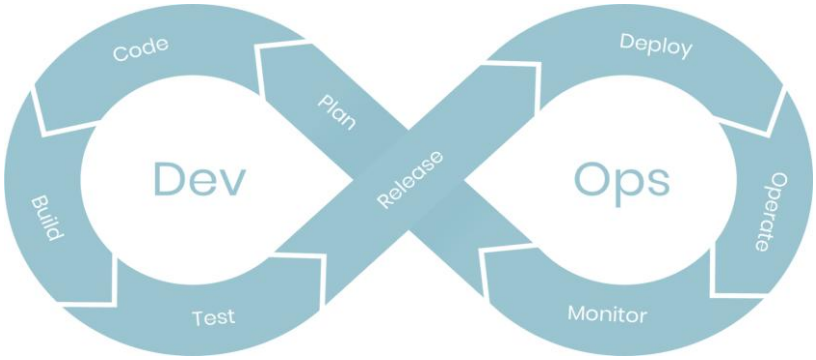


Рис. 7. DevOps модель [21]

Наприклад, при n=7 із врахуванням виразу (1), DevOps-модель можна представити таким чином:

$$SDLC_{DevOps} = \left\{\bigcup_{i=1}^{7} SDLC_i\right\} = \{SDLC_1, SDLC_2, \dots, SDLC_7\} = \{PLN, DEV, TES, INT, DPL, MTN, MON\}, \quad (12)$$

де $SDLC_1 = PLN$ – фаза планування, $SDLC_2 = DEV$ – фаза розроблення, $SDLC_3 = TES$ – фаза тестування, $SDLC_4 = INT$ – фаза інтегрування, $SDLC_5 = DPL$ – фаза розгортання, $SDLC_6 = MTN$ – фаза експлуатації, $SDLC_7 = MON$ – фаза моніторингу,

Враховуючи циклічність моделі (рис. 7), можна ввести множину відношень між фазами WFC і вираз (12) представити так:

$$DevOps = (SDLC_{DevOps}, WFC),$$
$$WFC = \begin{Bmatrix} (PLN, DEV), (DEV, TES), (TES, INT), (INT, DPL), \\ (DPL, MTN), (MTN, MON), (MON, PLN) \end{Bmatrix}. \quad (13)$$

Модель DevSecOps [9] є розвитком DevOps-моделі, що містить множину безпекових функцій. Таким чином, вибір моделі SDLC залежить від характеру проєкту, рівня регуляторних вимог, масштабу системи, критичності застосування та динаміки змін у середовищі. Для високоризикових або регламентованих систем доцільні формалізовані моделі з жорсткою трасованістю вимог, тоді як інноваційні IT-продукти в умовах швидкої ринкової еволюції ефективніше реалізовувати через гнучкі або гібридні підходи. У будь-якому випадку сучасна практика свідчить про необхідність поєднання структурованості з адаптивністю та обов'язкової інтеграції механізмів забезпечення якості й кібербезпеки протягом усього SDLC.

ОСНОВНА ЧАСТИНА ДОСЛІДЖЕННЯ

Запропонований метод інтегрування вимог кібербезпеки в SDLC реалізується в 5 етапів, зокрема, етап формалізації DevSecOps, етап ідентифікації вимог кібербезпеки, етап трансформації вимог у контрольні механізми, етап інтегрування контрольних механізмів у SDLC, етап верифікування та моніторинг виконання вимог кібербезпеки**.**

Далі більш детально розглянемо кожен із етапів з наведенням прикладів практичної реалізації.

Етап 1. Формалізація моделі DevSecOps. Введемо множину безпекових функцій $SECF$:

$$SECF = \{SREQ, STES, SVER, SCMP, SIMN\}, \quad (14)$$

де $SREQ$ – множина безпекових вимог, $STES$ – множина етапів безпекового тестування, $SVER$ – множина заходів безпекового верифікування, $SCMP$ – множина заходів безпекового комплаєнсу, $SIMN$ – множина заходів моніторингу інцидентів безпеки.

Враховуючи (12) – (14) модель DevSecOps матиме такий вигляд:

$$DevSecOps = (SDLC_{DevSecOps} \cup SECF, WFC^{ext}),$$
$$WFC^{ext} = WFC \cup WFC_{SECF},$$
$$WFC^{ext} = \begin{Bmatrix} (SREQ, DEV), (SREQ, TES), (STES, INT), (SVER, DPL), \\ (SCMP, MTN), (SIMN, MON) \end{Bmatrix}. \quad (15)$$

На наступному етапі необхідно ідентифікувати вимоги кібербезпеки, що будуть інтегруватися в SDLC.

Етап 2. Ідентифікація вимог кібербезпеки. Введемо множину вимог відповідно до [22] згідно принципів описаних вище, і відповідно до (15):

$$SREQ = \left\{ \bigcup_{i=1}^{n} SREQ_i \right\} = \{SREQ_1, SREQ_2, \dots, SREQ_n\}, \quad (16)$$

де $SREQ_i \subseteq SREQ (i = \overline{1,n})$ – вимоги кібербезпеки, визначені певним нормативним документом.

Наприклад, в разі використання міжнародного стандарту NIST 800-53 [11] при *n*=5 (з огляду на кількість базових безпекових контролів, не враховуючи додаткові), вираз (16) матиме вигляд:

$$NIST = \left\{ \bigcup_{i=1}^{5} NIST_i \right\} = \{NIST_1, NIST_2, \dots, NIST_5\} = \{AC, SC, CM, AU, IR\}, \quad (17)$$

де $NIST_1 = AC$ – це підмножина вимог контролю доступу, $NIST_2 = SC$ – це підмножина вимог безпеки зв'язку, $NIST_3 = CM$ – це підмножина вимог конфігурування системи, $NIST_4 = AU$ – це підмножина вимог аудиту та підзвітності, а $NIST_5 = IR$ – це підмножина вимог реагування на інциденти.

Для більш детального представлення вимог, враховуючи елементи (контролі) в структурі базових підмножин стандарту NIST 800-53 [11], вираз (17) можна представити наступним чином:

$$\begin{aligned} NIST = \left\{ \bigcup_{i=1}^{5} NIST_i \right\} &= \{NIST_1, NIST_2, \dots, NIST_5\} = \{AC, SC, CM, AU, IR\} \\ &= \{AC_1, AC_2, AC_3\} \cup \{SC_1, SC_2\} \cup \{CM_1, CM_2\} \cup \{AU_1, AU_2\} \\ &\cup \{IR_1, IR_2, IR_3\} = \\ &= \{AUTHORIZATION, AUTHENTICATION, ACCOUNTING\} \\ &\cup \{ENCRYPTION, CHANNEL\_SECURITY\} \\ &\cup \{CHANGES\_MANAGMENT, BASE\_CONFIG\} \\ &\cup \{LOGGING, MONITORING\} \\ &\cup \{DETECTION, RESPONSE, RECOVERY\} \end{aligned} \quad (18)$$

де $AC_1 = AUTHORIZATION$ – авторизація користувачів, $AC_2 = AUTHENTICATION$ – автентифікація користувачів, $AC_3 = ACCOUNTING$ – облікові записи, $SC_1 = ENCRYPTION$ – шифрування, $SC_2 = CHANNEL\_SECURITY$ – захист каналів передавання даних, $CM_1 = CHANGES\_MANAGMENT$ – управління змінами, $CM_2 = BASE\_CONFIG$ – базові конфігурування, $AU_1 = LOGGING$ – журналювання (логування), $AU_2 = MONITORING$ – моніторинг дій, $IR_1 = DETECTION$ – виявлення загроз, $IR_2 = RESPONSE$ – реагування, $IR_3 = RECOVERY$ – відновлення системи.

На наступному етапі необхідно трансформувати ідентифіковані вимоги кібербезпеки у контрольні механізми.

Етап 3. Трансформація вимог у контрольні механізми. Вимоги кібербезпеки визначені множиною (16), визначимо множину контрольних механізмів:

$$SMEC = \left\{ \bigcup_{j=1}^{m} SMEC_j \right\} = \{SMEC_1, SMEC_2, \dots, SMEC_m\}, \quad (19)$$

де $SMEC_j \subseteq SMEC (j = \overline{1, m})$ – контрольні механізми.

Тоді функцію трансформації вимог (16) у контрольні механізми (19) можна представити таким чином:

$$\begin{aligned} &f_{transformation}: SREQ \to SMEC, \\ &\forall\, SREQ_i \in SREQ\ \exists\, SMEC_j \in SMEC:\ SREQ_i \to SMEC_j \end{aligned} \quad (20)$$

Тобто абсолютно кожна вимога $SREQ_i$ трансформується в контрольні механізми $SMEC_j$.

Наприклад, вимога $AC_1 = AUTHORIZATION$ трансформується в механізми забезпечення авторизації $AUTHORIZATION\_MECHANISM$, вимога $SC_2 = CHANNEL\_SECURITY$ трансформується в засоби забезпечення захисту каналів передавання даних $CHANNEL\_SECURITY\_MECHANISM$, вимога $IR_2 = INCIDENT\_RESPONSE$ трансформується в інструменти реагування на кіберзагрози $INCIDENT\_RESPONSE\_MECHANISM$ і т.д.

Етап 4. Інтегрування контрольних механізмів у SDLC. Виразом (15) визначено множину етапів SDLC для моделі DevSecOps, відповідно DevSecOps-модель можна представити аналогічно (12):

$$SDLC_{DevSecOps} = \{DEV, TES, INT, DPL, MTN, MON\}, \tag{21}$$

Визначимо множину процесів DevSecOps:

$$PIPELINE_{DevSecOps} = \left\{\bigcup_{i=1}^{k} PIPELINE_i\right\} = \{PIPELINE_1, PIPELINE_2, \dots, PIPELINE_k\}, \tag{22}$$

де $PIPELINE_i \subseteq PIPELINE\left(i = \overline{1,k}\right)$ – процеси DevSecOps моделі.

Функцію інтегрування контрольних механізмів у SDLC на основі (19) – (22) можна представити наступним чином:

$$f_{integration}: SMEC \to SDLC \times PIPELINE, \quad \forall SMEC_j \in SMEC\ \exists\ (SDLC_i, PIPELINE_i): SMEC_j \to (SDLC_i, PIPELINE_i) \tag{23}$$

Таким чином, контрольний механізм $SMEC_j$ інтегрується в певний процес $PIPELINE_i$ конкретного етапу (фази) життєвого циклу розроблення ПЗ $SDLC_i$.

Для прикладу, контрольний механізм забезпечення авторизації $AUTHORIZATION\_MECHANISM$ інтегрується в процес персоналізації системи $PERSONNEL$ фази розроблення DEV, засоби забезпечення захисту каналів передавання даних $CHANNEL\_SECURITY\_MECHANISM$ інтегруються в процес захисту системи $SYSTEM\_PROTECTION$ тієї ж фази розроблення DEV, а інструменти реагування на кіберзагрози $INCIDENT\_RESPONSE\_MECHANISM$ інтегруються в процес управління інцидентами безпеки $INCIDENT\_MANAGEMENT$ фази моніторингу MON.

На наступному етапі відбувається верифікування та моніторинг виконання інтегрованих вимог кібербезпеки в SDLC.

Етап 5. Верифікування та моніторингу виконання вимог кібербезпеки. Множину метрик кібербезпеки можна визначити наступним чином:

$$CSM = \left\{\bigcup_{i=1}^{n} CSM_i\right\} = \{CSM_1, CSM_2, \dots, CSM_n\}, \tag{24}$$

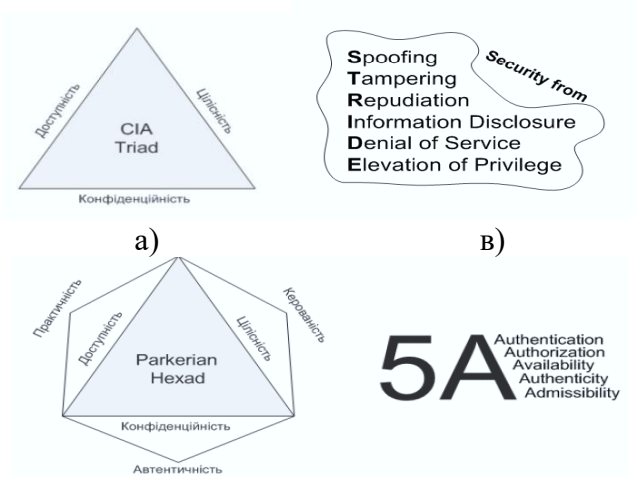
де $CSM_i \subseteq CSM\left(i = \overline{1,n}\right)$ – метрики безпеки (кібербезпеки).

Наприклад у відомій моделі кібербезпеки CIA [23] при n=3 вираз (24) матиме такий вигляд:

$$CSM_{CIA} = \left\{\bigcup_{i=1}^{3} CSM_i\right\} = \{CSM_1, CSM_2, CSM_3\} = \{CONFIDENTIALITY, INTEGRITY, AVAILABILITY\}, \quad (25)$$

де $CSM_1 = CONFIDENTIALITY$ – рівень (параметр) конфіденційності даних, $CSM_2 = INTEGRITY$ – рівень (параметр) цілісності даних, $CSM_3 = AVAILABILITY$ – рівень (параметр) доступності даних (інформаційних ресурсів).

Розширені моделі STRIDE, Parkerian Hexad та 5A, представлені на рис.8, згідно (24) можна відповідно формалізувати таким чином:



а) в)

б) г)

Рис. 8. Моделі кібербезпеки [24]: CIA (а), Parkerian Hexad (б), STRIDE (в) та 5A (г) [24]

1) при n=6 для моделі STRIDE вираз (24) матиме такий вигляд:

$$CSM_{STRIDE} = \left\{\bigcup_{i=1}^{6} CSM_i\right\} = \{CSM_1, CSM_2, \dots, CSM_6\} = \{SPOOF, TAMP, REPUD, ID, DoS, EoP\}, \quad (26)$$

де $CSM_1 = SPOOF$ – рівень захисту від підробки даних (Spoofing), $CSM_2 = TAMP$ – рівень захисту від несанкціонованого втручання в систему (Tampering), $CSM_3 = REPUD$ – рівень забезпечення неможливості відмови від авторства (Repudiation), $CSM_4 = ID$ – рівень захищеності від витоку й розголошення інформації з обмеженим доступом (Information disclosure), $CSM_5 = DoS$ – рівень захищеності від атак на відмову в обслуговуванні (Denial of service), $CSM_6 = EoP$ – рівень захищеності від розширення зловмисником прав доступу в системі (Elevation of privilege).

2) при n=6 для моделі Parkerian Hexad, що є фактичним розширенням моделі CIA (25), вираз (24) матиме такий вигляд:

$$CSM_{PH} = \left\{\bigcup_{i=1}^{6} CSM_i\right\} = \{CSM_1, CSM_2, \dots, CSM_6\} = \left\{\begin{matrix} CONFIDENTIALITY, INTEGRITY, AVAILABILITY, \\ POSSESSION, AUTHENTICITY, UTILITY \end{matrix}\right\}, \quad (27)$$

де $CSM_1 = CONFIDENTIALITY$ – рівень (параметр) конфіденційності даних, $CSM_2 = INTEGRITY$ – рівень цілісності даних, $CSM_3 = AVAILABILITY$ – рівень доступності

даних (інформаційних ресурсів), $CSM_4 = POSSESSION$ – рівень фізичного або логічного контролю даних, $CSM_5 = AUTHENTICITY$ – рівень аутентичності (підтвердження оригінальності авторства), $CSM_6 = UTILITY$ – рівень корисності та зручності використання даних.

3) при n=5 для моделі Б. Шнайєра 5А вираз (24) матиме такий вигляд:

$$CSM_{5A} = \left\{\bigcup_{i=1}^{5} CSM_i\right\} = \{CSM_1, CSM_2, \dots, CSM_5\} = \begin{Bmatrix} AUTHENTICATION, AUTHORIZATION, AVAILABILITY, \\ AUTHENTICITY, ADMISSIBILITY \end{Bmatrix}, \quad (28)$$

де $CSM_1 = AUTHENTICATION$ – рівень (параметр) забезпечення аутентифікації користувачів системи, $CSM_2 = AUTHORIZATION$ – рівень забезпечення авторизації користувачів системи, $CSM_3 = AVAILABILITY$ – рівень доступності даних (інформаційних ресурсів), $CSM_4 = AUTHENTICITY$ – рівень аутентичності (підтвердження оригінальності авторства), $CSM_5 = ADMISSIBILITY$ – рівень допустимості або прийнятності даних.

За результатами моніторингу, функцію оцінювання виконання вимог кібербезпеки (в продовження (20) та (23)), враховуючи (24), можна представити наступним чином:

$$f_{evaluation}: SMEC \to CSM \times VERIFICATION, \quad \forall SMEC_j \in SMEC \; \exists \, (CSM_i, VERIFICATION_i): SMEC_j \to (CSM_i, VERIFICATION_i), \quad (29)$$

де $VERIFICATION \subseteq [0,1]$ – параметр, що оцінює виконання контролів $SMEC_j$.

Ланцюг функціональних взаємодій методу можна представити у вигляді композиції відображень

$$SREQ \xrightarrow{f_{transformation}} SMEC \xrightarrow{f_{integration}} \langle SDLC \times PIPELINE \rangle \xrightarrow{f_{evaluation}} \langle CSM \times VERIFICATION \rangle . \quad (30)$$

Таким чином, відбувається перехід від вимог безпеки до верифікації метрик безпеки (через функції трансформації, інтегрування та оцінювання).

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

1) Використовуючи теорію множин, було формалізовано у загальному вигляді відомі моделі життєвого циклу розроблення ПЗ (класична каскадна модель, V-модель, інкрементальна модель, спіральна модель, Agile, DevOps, DevSecOps), що дало можливість сформувати уніфіковану модель SDLC.

2) Використовуючи уніфіковану модель SDLC, розроблено метод інтегрування вимог кібербезпеки в SDLC, що за рахунок формалізації DevSecOps, ідентифікації вимог кібербезпеки, трансформації вимог у контрольні механізми, інтегрування контрольних механізмів у SDLC, верифікування та моніторингу виконання вимог кібербезпеки у вигляді системи множин та відображень, дозволяє інтегрувати вимоги кібербезпеки в конкретну фазу (pipeline) життєвого циклу розроблення ПЗ, а також дає

можливість формалізованої оптимізації вибору контролів кібербезпеки залежно від контексту системи та ресурсних обмежень;

3) Практична цінність запропонованого методу полягає у можливості його використання для системного інтегрування вимог кібербезпеки в процеси розроблення ПЗ в організаціях, що створюють або експлуатують інформаційні системи критичної інфраструктури, хмарні сервіси та корпоративні інформаційно-комунікаційні системи.

4) Застосування розробленогометоду дозволяє: забезпечити узгодженість вимог кібербезпеки на всіх етапах SDLC; підвищити рівень автоматизації контролю кібербезпеки у DevSecOps pipeline; зменшити ризик пропуску критичних вимог кібербезпеки під час швидких ітерацій розроблення; підвищити обґрунтованість вибору засобів захисту інформації; скоротити витрати на усунення вразливостей за рахунок їх виявлення на ранніх етапах SDLC [19]; забезпечити можливість кількісного оцінювання рівня реалізації вимог кібербезпеки за допомогою метрик кібербезпеки; створити інструментальні засоби підтримки прийняття рішень у процесах DevSecOps.

5) Запропонований метод може бути використаний при проєктуванні захищених інформаційно-комунікаційних систем; аудиті процесів безпечної розробки ПЗ згідно стандартів ISO/IEC, NIST, PCI DSS, PSD2, GDPR, MITRE ATT&CK; створенні політик secure SDLC; впровадженні DevSecOps у державних та корпоративних IT-системах тощо**.**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moiseienko, V. M., & Antonenko, S. V. (2025). Research on the use of AI in the software development lifecycle. *Actual Problems of Automation and Information Technologies*, *29*, 293–305.
2. Delembovskyi, M., Markevych, M., & Korniichuk, B. (2024). Review of cybersecurity audit methodologies for compliance with standards. *Pidvodni Tekhnolohii*, *1*(14), 71–74. https://doi.org/10.32347/uwt.2024.14.1206
3. Zhuravchak, A., & Piskozub, A. (2025). Analysis of machine learning methods for automating penetration testing. *Cybersecurity: Education, Science, Technique*, *3*(27), 54–62. https://doi.org/10.28925/2663-4023.2025.27.711
4. Saini, J., & Bansal, A. (2024). Automated penetration testing: Machine learning approach. In *Symposium on Computing Intelligent Systems (SCI)* (Vol. 3682, pp. 113–125).
5. Foros, A. V. (2009). Information security as a component of national security of Ukraine. *Pravova Derzhava*, *11*, 222–226.
6. Khari, M., Vaishali, & Kumar, P. (2016). Embedding security in software development lifecycle (SDLC). In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 2182–2186).
7. IT Notes. (n.d.). Software development lifecycle (SDLC). https://www.it-notes.wiki/other/software-development-lifecycle
8. Manjeti, V., Penumajji, S., Patlolla, S., et al. (2025). Enhancing security in SDLC with DevOps tools and practices. In *2025 International Conference on Next Generation of Green Information and Emerging Technologies (GIET)* (pp. 1–5). https://doi.org/10.1109/GIET65294.2025.11234805
9. Bhardwaj, A., Anugula, P., et al. (2025). Zero trust CI/CD pipeline: A blueprint for secure software delivery in modern DevSecOps. In *2025 IEEE Uttar Pradesh Section WIE International Conference* (pp. 233–237). https://doi.org/10.1109/UPWIECON67212.2025.11390387
10. International Organization for Standardization. (2022). *ISO/IEC 27001:2022—Information security management systems—Requirements*.
11. National Institute of Standards and Technology. (2020). *Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5)*. https://doi.org/10.6028/NIST.SP.800-53r5
12. Hancock, S. (2025). *PCI DSS version 4.0.1: A guide to the payment card industry data security standard*. Packt Publishing.
13. Wodo, W., & Stygar, D. (2021). PSD2-compliant hardware token for digital banking. In *62nd International Scientific Conference on Information Technology and Management Science (ITMS)* (pp. 1–6).

14. IT Governance Privacy Team. (2025). *EU General Data Protection Regulation (GDPR): An implementation and compliance guide*. Packt Publishing.
15. Tsai, W., Luo, J.-N., & Chou, C.-L. (2025). Integrating tree structures with the MITRE ATT&CK framework for APT detection. In *2025 9th International Conference on Cryptography, Security and Privacy (CSP)* (pp. 139–143). https://doi.org/10.1109/CSP66295.2025.00031
16. TechnologyAdvice. (n.d.). What is waterfall project management? https://technologyadvice.com/blog/project-management/what-is-waterfall-project-management
17. Teaching Agile. (n.d.). V-model in software development. https://teachingagile.com/sdlc/models/v-model
18. Guru99. (n.d.). Incremental model in SDLC. https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html
19. Teaching Agile. (n.d.). Spiral model. https://teachingagile.com/sdlc/models/spiral
20. InterviewBit. (n.d.). Agile model. https://www.interviewbit.com/blog/agile-model
21. BETSOL. (n.d.). What is DevOps? https://www.betsol.com/blog/what-is-devops
22. Skurativskyi, A. (2025). Method for managing cybersecurity requirements in software implementation in business. *Information Security*, *3*, 145–162.
23. Seol, J., Deuja, J., et al. (2025). A quantitative study across the CIA triad and performance in blockchain-based crypto-space. In *2025 7th International Conference on Blockchain Computing and Applications (BCCA)* (pp. 161–168). https://doi.org/10.1109/BCCA66705.2025.11229817
24. Kharchenko, V., Korchenko, O., & Hnatiuk, S. (2017). Multilevel data model for compliance with cybersecurity regulatory requirements in civil aviation. *Information Protection, 19*(1), 95–104. https://doi.org/10.18372/2410-7840.19.11499
25. Raj, G., Singh, D., & Bansal, A. (2014). Analysis for security implementation in SDLC. In *2014 5th International Conference – Confluence* (pp. 221-226). https://doi.org/10.1109/CONFLUENCE.2014.6949376

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (32), 2026
ISSN 2663 – 4023

Sergiy Gnatyuk
Doctor of Technical Sciences, Professor, Vice-Rector for Scientific Research and Technology Transfer
State University "Kyiv Aviation Institute", Kyiv, Ukraine
ORCID: 0000-0003-4992-0564
s.gnatyuk@kai.edu.ua

Zarina Poberezhna
Doctor of Economic Sciences, Professor, Professor of the Department of National Security and Entrepreneurship
State University "Kyiv Aviation Institute", Kyiv, Ukraine
ORCID: 0000-0001-6245-038X
zarina.poberezhna@npp.kai.edu.ua

Anatolii Skurativskyi
Postgraduate Student, Department of Computer Information Technologies
State University "Kyiv Aviation Institute", Kyiv, Ukraine
ORCID: 0009-0000-0566-2394
a.skurativskyi@gmail.com

METHOD OF INTEGRATION OF CYBERSECURITY REQUIREMENTS INTO THE SOFTWARE DEVELOPMENT LIFECYCLE

Abstract. In modern conditions, it is important not only to develop software to protect information systems and data, but also to integrate security functions (cybersecurity) into the phases of the software development life cycle. In view of this, the work formalized in a general form the well-known models of the software development life cycle and formed a unified SDLC model. Also, a method for integrating cybersecurity requirements into the SDLC was developed, which allows integrating cybersecurity requirements into a specific phase (pipeline) of the software development life cycle in accordance with the DevSecOps model, and also allows for formal optimization of the choice of cybersecurity controls depending on the system context and resource constraints. The results obtained can be used for the systematic integration of cybersecurity requirements (according to regulatory documents ISO/IEC, NIST, PCI DSS, PSD2, GDPR, MITRE ATT&CK) into software development processes in organizations that create or operate critical infrastructure information systems, cloud services, and corporate information and communication systems.

Keywords: cybersecurity; software development life cycle; SDLC; DevSecOps; threat modeling; security; cybersecurity requirements.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. Moiseienko, V. M., & Antonenko, S. V. (2025). Research on the use of AI in the software development lifecycle. *Actual Problems of Automation and Information Technologies, 29*, 293–305.
2. Delembovskyi, M., Markevych, M., & Korniichuk, B. (2024). Review of cybersecurity audit methodologies for compliance with standards. *Pidvodni Tekhnolohii, 1*(14), 71–74. https://doi.org/10.32347/uwt.2024.14.1206
3. Zhuravchak, A., & Piskozub, A. (2025). Analysis of machine learning methods for automating penetration testing. *Cybersecurity: Education, Science, Technique, 3*(27), 54–62. https://doi.org/10.28925/2663-4023.2025.27.711
4. Saini, J., & Bansal, A. (2024). Automated penetration testing: Machine learning approach. In *Symposium on Computing Intelligent Systems (SCI)* (Vol. 3682, pp. 113–125).
5. Foros, A. V. (2009). Information security as a component of national security of Ukraine. *Pravova Derzhava, 11*, 222–226.
6. Khari, M., Vaishali, & Kumar, P. (2016). Embedding security in software development lifecycle (SDLC). In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)* (pp. 2182–2186).

7. IT Notes. (n.d.). Software development lifecycle (SDLC). https://www.it-notes.wiki/other/software-development-lifecycle
8. Manjeti, V., Penumajji, S., Patlolla, S., et al. (2025). Enhancing security in SDLC with DevOps tools and practices. In *2025 International Conference on Next Generation of Green Information and Emerging Technologies (GIET)* (pp. 1–5). https://doi.org/10.1109/GIET65294.2025.11234805
9. Bhardwaj, A., Anugula, P., et al. (2025). Zero trust CI/CD pipeline: A blueprint for secure software delivery in modern DevSecOps. In *2025 IEEE Uttar Pradesh Section WIE International Conference* (pp. 233–237). https://doi.org/10.1109/UPWIECON67212.2025.11390387
10. International Organization for Standardization. (2022). *ISO/IEC 27001:2022—Information security management systems—Requirements*.
11. National Institute of Standards and Technology. (2020). *Security and privacy controls for information systems and organizations (NIST SP 800-53 Rev. 5)*. https://doi.org/10.6028/NIST.SP.800-53r5
12. Hancock, S. (2025). *PCI DSS version 4.0.1: A guide to the payment card industry data security standard*. Packt Publishing.
13. Wodo, W., & Stygar, D. (2021). PSD2-compliant hardware token for digital banking. In *62nd International Scientific Conference on Information Technology and Management Science (ITMS)* (pp. 1–6).
14. IT Governance Privacy Team. (2025). *EU General Data Protection Regulation (GDPR): An implementation and compliance guide*. Packt Publishing.
15. Tsai, W., Luo, J.-N., & Chou, C.-L. (2025). Integrating tree structures with the MITRE ATT&CK framework for APT detection. In *2025 9th International Conference on Cryptography, Security and Privacy (CSP)* (pp. 139–143). https://doi.org/10.1109/CSP66295.2025.00031
16. TechnologyAdvice. (n.d.). What is waterfall project management? https://technologyadvice.com/blog/project-management/what-is-waterfall-project-management
17. Teaching Agile. (n.d.). V-model in software development. https://teachingagile.com/sdlc/models/v-model
18. Guru99. (n.d.). Incremental model in SDLC. https://www.guru99.com/what-is-incremental-model-in-sdlc-advantages-disadvantages.html
19. Teaching Agile. (n.d.). Spiral model. https://teachingagile.com/sdlc/models/spiral
20. InterviewBit. (n.d.). Agile model. https://www.interviewbit.com/blog/agile-model
21. BETSOL. (n.d.). What is DevOps? https://www.betsol.com/blog/what-is-devops
22. Skurativskyi, A. (2025). Method for managing cybersecurity requirements in software implementation in business. *Information Security*, *3*, 145–162.
23. Seol, J., Deuja, J., et al. (2025). A quantitative study across the CIA triad and performance in blockchain-based crypto-space. In *2025 7th International Conference on Blockchain Computing and Applications (BCCA)* (pp. 161–168). https://doi.org/10.1109/BCCA66705.2025.11229817
24. Kharchenko, V., Korchenko, O., & Hnatiuk, S. (2017). Multilevel data model for compliance with cybersecurity regulatory requirements in civil aviation. *Information Protection*, *19*(1), 95–104. https://doi.org/10.18372/2410-7840.19.11499
25. Raj, G., Singh, D., & Bansal, A. (2014). Analysis for security implementation in SDLC. In *2014 5th International Conference – Confluence* (pp. 221-226). https://doi.org/10.1109/CONFLUENCE.2014.6949376

Отримано редакцією журналу / Received: 26.01.26
Прорецензовано / Revised: 15.02.26
Схвалено до друку / Accepted: 26.03.26