ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Архітектура сервісу онлайн-моніторингу для захисту банківських активів клієнта»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Олександр ГОРБАР*
(підпис) (Ім'я, ПРІЗВИЩЕ здобувача)

Виконав:
здобувач вищої освіти
група ІСДМ-64
Олександр ГОРБАР
(Ім'я, ПРІЗВИЩЕ)

Керівник:
науковий ступінь, вчене звання
Віктор САГАЙДАК
к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання
_____
(Ім'я, ПРІЗВИЩЕ)

Київ – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ

_____Каміла СТОРЧАК
«____» _____20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ СТУДЕНТУ
Горбар Олександр Едуардович
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Архітектура сервісу онлайн-моніторингу для захисту банківських активів клієнта

керівник кваліфікаційної роботи Сагайдак Віктор Анатолійович, к.т.н., доцент
(ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «15» жовтня 2024 року № 320.

2. Строк подання кваліфікаційної роботи: 27 грудня 2024 року.

3. Вихідні дані до кваліфікаційної роботи: архітектура програмного забезпечення; стандарти систем безпеки програмного забезпечення;
науково-технічна література з питань, пов'язаних з наукою про дані.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз вимог до програмного каркасу безпеки та їх відповідність стандартам.
2. Розробка структури уніфікованого програмного каркасу.
3. Оцінка та валідація розробленого каркасу.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання: 15 жовтня 2024 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявної науково-технічної літератури	15.10 – 23.10.24	
2	Аналіз стандартів безпеки та їх інтеграції в життєвий цикл розробки (SDLC)	24.10 – 01.11.24	
3	Дослідження методів картографування безпекових стандартів	02.11 – 07.11.24	
4	Розробка структури уніфікованого програмного каркасу	08.11 – 16.11.24	
5	Проведення експертної оцінки розробленого каркасу	17.11 – 21.11.24	
6	Оформлення роботи: вступ, висновки, реферат	22.11 – 11.12.24	
7	Оформлення роботи: вступ, висновки, реферат	12.12 – 20.12.24	
8	Розробка демонстраційних матеріалів	21.12 – 25.12.24	

Здобувач вищої освіти _____ *Олександр ГОРБАР*
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник роботи
кваліфікаційної роботи _____ Віктор САГАЙДАК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 85 стор., 10 рис., 8 табл., 64 джерел.

Мета роботи – розробка уніфікованого програмного каркасу безпеки програмного забезпечення для інтеграції стандартів безпеки у життєвий цикл розробки (SDLC) з урахуванням потреб фінансової індустрії.

Об'єкт дослідження – процес побудови програмного каркасу для забезпечення інформаційної безпеки.

Предмет дослідження – інтеграція стандартів OWASP ASVS та CIS Controls у програмний каркас для SDLC.

Короткий зміст роботи: У роботі проведено аналіз стандартів безпеки програмного забезпечення, таких як OWASP ASVS, CIS Controls, ISO 27001, та їх застосування для забезпечення відповідності регуляторним вимогам фінансової індустрії. Досліджено методи картографування безпекових стандартів, визначено ключові аспекти для їх інтеграції у SDLC. Розроблено структуру уніфікованого програмного каркасу безпеки, що відповідає специфічним потребам галузі, а також оцінено його ефективність за допомогою експертного аналізу. На основі отриманих результатів сформульовано рекомендації щодо вдосконалення та застосування каркасу для підвищення рівня безпеки програмного забезпечення.

КЛЮЧОВІ СЛОВА: ПРОГРАМНИЙ КАРКАС, ІНФОРМАЦІЙНА БЕЗПЕКА, SDLC, OWASP ASVS, CIS CONTROLS, ФІНАНСОВА ІНДУСТРІЯ, СТАНДАРТИ БЕЗПЕКИ.

ABSTRACT

Text part of the master`s qualification work: 85 pages, 10 pictures, 8 tables, 24 64 sources.

The purpose of the work the development of a unified software security framework for integrating security standards into the software development lifecycle (SDLC), considering the needs of the financial industry.

Object of research the process of building a security framework for ensuring information security.

Subject of research the integration of OWASP ASVS and CIS Controls standards into the security framework for SDLC.

Summary of the work: The work includes an analysis of software security standards, such as OWASP ASVS, CIS Controls, and ISO 27001, and their application to ensure compliance with regulatory requirements in the financial industry. Methods for mapping security standards were investigated, and key aspects for their integration into the SDLC were identified. A structure of a unified software security framework was developed to meet the specific needs of the industry, and its effectiveness was evaluated through expert analysis. Based on the results, recommendations were formulated for improving and applying the framework to enhance software security.

KEYWORDS: SECURITY FRAMEWORK, INFORMATION SECURITY, SDLC, OWASP ASVS, CIS CONTROLS, FINANCIAL INDUSTRY, SECURITY STANDARDS.

ЗМІСТ

ВСТУП

Актуальність теми. Ландшафт кібербезпеки інформаційних технологій (IT) постійно змінюється, і організації стикаються з суттєвими викликами у дотриманні численних стандартів і нормативів безпеки. Вимоги до відповідності можуть надходити як від клієнтів, так і від глобальних регуляцій, таких як Загальний регламент захисту даних (GDPR) Європейського Союзу (ЄС). Наслідки для організацій, які не дотримуються належних регуляцій, можуть бути катастрофічними. Ці наслідки можуть варіюватися від незначних штрафів або санкцій, пошкодження репутації організації до серйозних випадків, таких як тюремне ув'язнення[32]. Дотримання регіональних та національних регуляцій є вимогою закону, тому його завжди слід дотримуватись. Хоча це не завжди прямо вимагається законом, різні стандарти безпеки сприяють дотриманню цих регуляцій. [56] Безпекові вимоги та стандарти також важливі, оскільки вони пропонують гарну основу для створення процесів управління інформаційною безпекою, політик та інших активностей, що позитивно впливає на організацію.[40]

Однак дотримання та впровадження кожного стандарту безпеки та рекомендацій може стати складним завданням. Більшість поширених стандартів суттєво перетинаються між собою. Через велику кількість різних стандартів безпеки організації не можуть дотримуватися їх усіх, тому зазвичай вибирають лише декілька. Вибір і дотримання обраних стандартів впливає на всі аспекти розробки програмного забезпечення організацій: від проєктування до підтримки. Впровадження цих стандартів і рекомендацій повинно бути максимально спрощеним, щоб їх можна було коректно інтегрувати в життєвий цикл розробки програмного забезпечення організації. Саме тому уніфікований програмний каркас, який об'єднує різні стандарти в одну систему, є необхідним, адже він оптимізує процес планування безпекових вимог для організацій протягом усього життєвого циклу програмного забезпечення.

Більшість поширених стандартів безпеки вже можна співвіднести один з одним. Однак досі існує потреба у відповідному узгодженні для фреймворків на рівні застосунків, таких як Стандарт перевірки безпеки застосунків (OWASP ASVS), і стандартів організаційного рівня, таких як ISO 27001; NIST 800-53 та Критичні контрольні заходи безпеки Центру Інтернет-безпеки (CIS CSC).

Мета роботи – об’єднання рекомендацій з безпеки на рівні застосунків зі стандартами на рівні інфраструктури в єдиний уніфікований програмний каркас, щоб з’ясувати, як вони пов’язані та можуть бути використані в життєвому циклі розробки програмного забезпечення (SDLC) організацій. На основі цього програмного каркасу можна буде спрямовувати процес розробки для використання належних контрольних заходів безпеки та рекомендацій на кожному етапі розробки. Зрештою, програмний каркас буде слугувати контрольним списком та керівництвом для зацікавлених сторін, щоб вони знали, які контрольні заходи та зміцнення повинні бути додані до програмного забезпечення та середовища для відповідності стандартам на кожному етапі розробки.

Об’єкт дослідження – процес побудови програмного каркасу безпеки програмного забезпечення.

Предмет дослідження – архітектура безпеки програмного забезпечення.

Методи дослідження. Під час виконання завдань магістерської кваліфікаційної роботи були використані методи елементів системного аналізу, методи теоретичного дослідження, імітаційне моделювання.

Мотивацією для цієї роботи було те, що наразі не існує загального програмного каркасу безпеки програмного забезпечення, який би поєднував стандарти безпеки з різними етапами розробки програмного забезпечення, яку організації могли б використовувати у своїх процесах. Різні стандарти пропонують інструкції для впровадження та взаємозв'язок із іншими стандартами. Проте відсутні рекомендації щодо того, як їх слід інтегрувати в процеси під час розробки програмного забезпечення. Це й стало поштовхом до створення раніше

згаданого програмного каркасу. Водночас можна буде оцінити, наскільки добре стандарти, обрані для цього програмного каркасу, відповідають вимогам галузі.

Наукова новизна одержаних результатів. Запропоновано нову структуру SDLC із додатковою фазою «Загальні вимоги», що враховує організаційні аспекти безпеки. Доведено відповідність каркасу вимогам Європейського банківського управління, забезпечено його застосовність у фінансовій галузі. Вперше продемонстровано ефективність мапінгу вимог безпеки для спрощення відповідності стандартам. Каркас сприяє покращенню безпеки програмного забезпечення, стандартизує процеси, підвищує обізнаність розробників і підтримує відповідність регуляторним вимогам.

Практична значущість одержаих результатів. Розроблений програмний каркас спрощує управління вимогами безпеки, забезпечує відповідність регуляторним стандартам і підвищує якість захисту застосунків на всіх етапах SDLC. Його можна використовувати для оптимізації процесів розробки, навчання персоналу, моніторингу відповідності та демонстрації виконання вимог безпеки клієнтам і аудиторам. Запропоновані рішення зменшують ризики кіберзагроз і сприяють розвитку культури інформаційної безпеки в організаціях.

Апробація результатів магістерської роботи. Базові результати магістерської роботи опубліковано в збірнику тез II Всеукраїнській науково-технічній конференції "Технологічні горизонти: дослідження та застосування інформаційних технологій для технологічного прогресу України і світу".

1. АНАЛІЗ ТЕХНОЛОГІЇ ПРОГРАМНОГО КАРКАСУ БЕЗПЕКИ

1.1. Цілі та обмеження

Як зазначалося раніше, ця дипломна робота спрямована на розробку технічного програмного каркасу безпеки, який інтегрує рекомендації з безпеки на рівні застосунків із ширшими стандартами на рівні інфраструктури в єдиний уніфікований програмний каркас, що може бути використаний у життєвому циклі розробки програмного забезпечення (SDLC) організацій. Таким чином, цей програмний каркас вирішить проблему використання різних стандартів безпеки протягом усього життєвого циклу програмного забезпечення — від етапу визначення вимог до етапу підтримки. Запропонований програмний каркас може слугувати як керівництво для оцінки того, які контрольні заходи та дії потрібні для різних застосунків залежно від критичності безпеки на кожному етапі розробки. Це означає, що фахівці з безпеки можуть використовувати програмний каркас як основу для налаштування необхідних контрольних заходів і політик для середовищ, а розробники — як посібник у процесі розробки програмного забезпечення.

Другою метою роботи є аналіз того, наскільки добре рекомендації у запропонованому програмному каркасі узгоджуються з вимогами безпеки IT-фінансової індустрії, а також як різні стандарти безпеки, що використовуються в інших контекстах, можуть бути поєднані. Це дозволить отримати цінну інформацію про те, як різні рекомендації з безпеки можуть бути уніфіковані для задоволення специфічних потреб галузі. Тема дипломної роботи також є актуальною, оскільки кіберінциденти наразі є найважливішим бізнес-ризиком у світі. [25]

Дослідження також має на меті зіставити різні програмні каркаси, розроблені для інших сценаріїв використання. Запланований програмний каркас повинен пропонувати рекомендації як для інфраструктурних, так і для корпоративних вимог, а також узгоджувати ці потреби з критичністю безпеки

застосунків. Тому важливо з’ясувати, наскільки добре ці рекомендації та стандарти узгоджуються між собою і скільки зіставлення можна здійснити.

Програмний каркас має бути сумісним з іншими стандартами, тобто до нього можна буде додавати нові стандарти та політики. Розробка програмного каркасу базується на конкретних вимогах організації та галузевих рекомендаціях, тому він буде інтегрований у життєвий цикл розробки програмного забезпечення організації. Оцінка програмного каркасу здійснюватиметься на основі галузевих стандартів безпеки, попередніх наукових досліджень і того, наскільки добре він відповідає потребам організації.

Оскільки тема програмного забезпечення та кібербезпеки є дуже широкою, а різні організації створюють різні стандарти й рекомендації, для збереження доцільності обсягу дипломної роботи необхідно встановити обмеження. Таким чином, обмеження цієї роботи є такими: буде розроблено лише технічний програмний каркас, без фактичного впровадження в умовах виробництва. Оцінка артефакту проводитиметься лише текстово, без аналізу його реалізації в реальних умовах. Також усі рекомендації та контрольні заходи, зазначені у програмному каркасі, можуть не підходити для всіх моделей доставки застосунків. Для зменшення складності програмного каркасу у роботі буде використано лише два конкретні стандарти безпеки, визначені компанією.

1.2. Базові вимоги та обмеження

Практичне впровадження цієї дипломної роботи здійснюється для Електронної фінансової інституції (EMI), компанії, яка надає програмні рішення та консультаційні послуги для фінансового сектора.[17] Програмний каркас розроблений для внутрішнього використання компанією як частина її системи управління інформаційною безпекою та документації життєвого циклу розробки програмного забезпечення (SDLC). Поточна версія не буде використовуватися як комерційний аргумент для потенційних клієнтів. Вона слугуватиме керівництву

для загальної розробки програмного забезпечення і стане частиною внутрішнього посібника організації з розробки програмного забезпечення. Таким чином, запропонований програмний каркас буде використовуватися для кожного застосунку, який організація створюватиме в майбутньому. Хоча створений програмний каркас може також бути застосований до клієнтського інтерфейсу, це не входить до сфери цієї дипломної роботи. Жодна реальна реалізація каркасу не буде виконана, але його оцінюватимуть відповідні співробітники організації на основі його корисності.

Виходячи з потреб організації, запропонований програмний каркас має відповідати існуючим інструкціям компанії щодо програмного забезпечення та кібербезпеки. Таким чином, у запропонованому каркасі використовуються OWASP ASVS і CIS Controls, які будуть інтегровані у внутрішній SDLC організації. З-поміж багатьох різних стандартів і рекомендацій безпеки CIS та OWASP ASVS були обрані через їх популярність у Європі, особливо у фінансовому секторі, а також через їхню відповідність потребам організації. OWASP-рекомендації є основними стандартами, з якими зазвичай хочуть, щоб партнери дотримувалися під час розробки програмного забезпечення у фінтех-індустрії Європи. Стандарти безпеки Національного інституту стандартів і технологій (NIST) пропонують схожі рекомендації, але вони частіше використовуються у Сполучених Штатах і багато з них створені за дорученням уряду США.[18] Однак слід зазначити, що більшість організацій, які використовують програмний каркас кібербезпеки NIST, також відповідають вимогам CIS Controls і можуть використовувати їх спільно. [14]

1.3. Запитання до дослідження (Research questions)

У дипломній роботі буде розглянуто два дослідницькі питання (RQ). Спочатку представлено перше дослідницьке питання та обговорено його важливість для роботи.

RQ1: Як можна розробити програмний каркас безпеки, що інтегрує рекомендації з безпеки на рівні застосунків і стандарти на рівні інфраструктури з SDLC організації?

Стандарти, які планується використовувати в програмному каркасі, призначені для різних сценаріїв. Один з них стосується рекомендацій щодо безпеки застосунків, а інший — контролю на рівні інфраструктури. Тому важливо виявити основні подібності між ними та зрозуміти, як вони можуть бути інтегровані в SDLC організації. В іншому випадку впровадження програмного каркасу стає громіздким. Крім того, необхідно з'ясувати, як організація може використовувати цей каркас і які переваги він може принести загальному ландшафту безпеки.

RQ2: Наскільки добре рекомендації запропонованого програмного каркасу враховують специфічні вимоги до безпеки в IT-фінансовій індустрії?

Під час розробки програмного забезпечення для фінансового сектора організації мають дотримуватись численних регуляцій і вимог. Оцінка того, наскільки програмний каркас узгоджується з цими вимогами, є критично важливою для його успіху. Це також може виявити недоліки стандартів, які використовуються в каркасі, і показати, як ці стандарти відхиляються від загальноприйнятих стандартів безпеки або відповідають вимогам галузі. Оскільки фінансові дані є надзвичайно чутливими, а регуляторне середовище — суворим, розуміння того, наскільки добре запропонований каркас відповідає цим специфічним вимогам, має вирішальне значення.

1.4. Безпекові заходи Центру Інтернет-безпеки (CIS) Controls

Контрольні заходи CIS є одними з найпоширеніших програмних каркасів безпеки, які використовуються в IT-індустрії.[40; 63; 13] Це список критичних заходів безпеки, розроблений Центром Інтернет-безпеки (Center for Internet Security). Ці заходи описуються як «приписи, пріоритетні та спрощені найкращі практики, які можна використовувати для зміцнення вашого кібербезпекового стану».[31] Вони можуть бути використані для дотримання різних галузевих регуляцій і спрощення захисту програмного забезпечення від загроз.[31] Контрольні заходи CIS призначені бути базовою основою безпеки організації та допомагають пріоритизувати увагу на найбільш критичних аспектах у сфері безпеки. [31]

Остання версія, актуальна на момент написання цієї роботи, — це восьма ітерація цих заходів. У цій версії є 18 різних заходів, кожен із яких зосереджений на окремих аспектах безпеки застосунків. Контрольні заходи поділяються на три групи впровадження (IG1, IG2, IG3), і ці групи можуть бути пов’язані з розміром організації та її експертизою у сфері безпеки. Кожна група впровадження базується на попередній, тобто IG3 включає всі заходи CIS, що містяться в IG1 і IG2. Вибір групи впровадження залежить від підприємства. Нижче наведено основну логіку груп впровадження (див. Рисунок 1). [31]

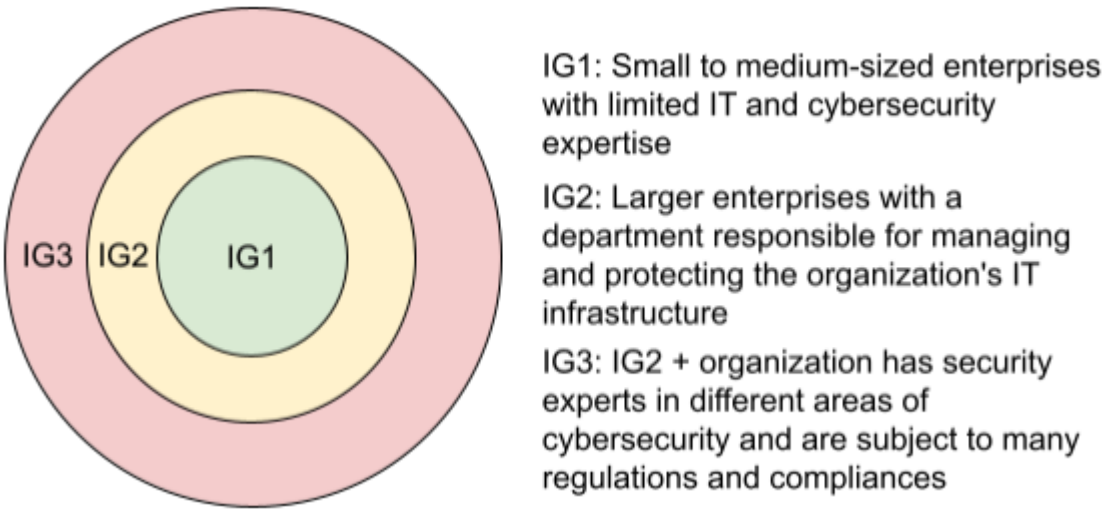


Рисунок 1. Логіка груп впровадження [31]

Група IG1 призначена переважно для малих та середніх підприємств, які мають обмежений рівень експертизи у сфері інформаційних технологій (IT) та кібербезпеки. Заходи безпеки, зазначені в цій групі, спрямовані на захист від «загальних нетаргетованих атак» і «розроблені для роботи з комерційним програмним та апаратним забезпеченням, що доступне на ринку» (COTS). [31]

Група IG2 орієнтована на великі підприємства, які мають відділ, відповідальний за управління та захист IT-інфраструктури організації. Заходи цієї групи спрямовані на допомогу відділу безпеки у контролі за зростаючою складністю операцій. Організації цієї групи можуть уже мати потреби у дотриманні регуляцій, з якими допоможуть впоратися додаткові заходи. [31]

Група IG3 є найвищою та найширшою групою впровадження, яка включає всі заходи безпеки CIS. У цій групі організація має експертів із різних галузей кібербезпеки. Такі організації володіють «даними та активами, що містять чутливу інформацію та підпадають під регуляторний контроль і дотримання стандартів». Логіка полягає в тому, що чим більша організація, тим вищий рівень впровадження. [31]

Ці заходи безпеки категоризовані за типом активів (пристрої, застосунки, дані, мережі, користувачі) та за їхньою функцією безпеки (ідентифікація,

реагування, виявлення, захист, відновлення). Загалом у рамках усіх контрольних заходів CIS існує 153 заходи. Тип активів визначає, на які саме активи впливає захід, а категорія функцій безпеки визначає, як впровадження цих заходів сприятиме підвищенню загальної інфраструктури безпеки організації. Заходи безпеки в програмному каркасі створені для того, щоб запропонувати керівництво, яке охоплює найпоширеніші сфери безпеки, такі як захист даних, управління обліковими записами, контроль доступу та безпека застосунків. Заходи розроблені як «високоефективні захисні дії, які забезпечують обов'язкову базу, з якої мають починати підприємства». [31]

Наприклад, захід CIS 3.10: Шифрування чутливих даних під час передачі входить до категорії заходів із захисту даних. Для його виконання організації повинні шифрувати чутливі дані під час передачі, використовуючи відповідні протоколи, такі як Transport Layer Security (TLS) і Open Secure Shell (OpenSSH). Цей захід призначений для рівнів IG2 та IG3 і належить до категорії «Захист». Це означає, що більшість організацій повинні впроваджувати ці заходи у своїх процесах розробки для забезпечення захисту і зменшення ризиків кіберзагроз. [31]

Перелік 18 заходів безпеки CIS Controls версії 8:

1. Інвентаризація та контроль корпоративних активів
2. Інвентаризація та контроль програмних активів
3. Захист даних
4. Безпечна конфігурація корпоративних активів і програмного забезпечення
5. Управління обліковими записами
6. Управління контролем доступу
7. Постійне управління вразливостями
8. Управління журналами аудиту
9. Захист електронної пошти та веббраузера

10. Захист від шкідливого програмного забезпечення
11. Відновлення даних
12. Управління мережею
13. Моніторинг і захист мережі
14. Навчання безпеки та розвиток навичок
15. Управління постачальниками послуг
16. Безпека програмного забезпечення застосунків
17. Управління реагуванням на інциденти
18. Тестування на проникнення (penetration testing)

На відміну від інших поширених програмних каркасів, таких як NIST CSF або ISO 27001, заходи CIS не допомагають вирішувати питання управління ризиками або пропонувати керівництво для аналізу ризиків. Їхня основна функція полягає у наданні рекомендацій, які допомагають організаціям зміцнити технічну інфраструктуру та одночасно зменшити ризики безпеки. Крім того, CIS Foundation пропонує сертифікацію для організацій, що підвищує їхню репутацію у світових переговорах. Програмний каркас визнаний багатьма авторитетами у світі, зокрема урядом США, Європейським інститутом телекомунікаційних стандартів (ETSI), Національною асоціацією губернаторів (NGA) та Центром захисту національної інфраструктури Великобританії (CPNI). [31]

Подібно до багатьох інших стандартів і програмних каркасів, заходи CIS можна використовувати та інтегрувати зі стандартами, такими як серія ISO 27000, стандарти NIST та регуляції, зокрема GDPR і FISMA. Відповідно до офіційної документації CIS, ці контрольні заходи «не є заміною існуючих регуляцій, стандартів відповідності або схем авторизації» і рекомендуються до використання разом з іншими стандартами.

Деякі існуючі заходи вже адаптовані для хмарних середовищ, наприклад, для Azure, за допомогою CIS Microsoft Azure Foundations Benchmark. Різні CIS Benchmarks призначені для надання більш детальних інструкцій і найкращих

практик щодо безпечної конфігурації систем і середовищ. Ці рекомендації також визнані великими організаціями, такими як Microsoft. [31]

1.5. Стандарт перевірки безпеки застосунків OWASP (ASVS)

Стандарт OWASP ASVS від OWASP Foundation розроблений для того, щоб допомогти організаціям створювати та підтримувати безпечні застосунки. Основна мета стандарту полягає у «нормалізації діапазону охоплення та рівня строгості, доступного на ринку під час перевірки безпеки вебзастосунків із використанням комерційно придатного відкритого стандарту». [45]Його основний акцент зроблено на визначенні функціональних і нефункціональних заходів безпеки, що використовуються під час проєктування, розробки та тестування вебзастосунків.

Остання версія OWASP ASVS — це четверта ітерація стандарту, яка включає 14 категорій. [45]

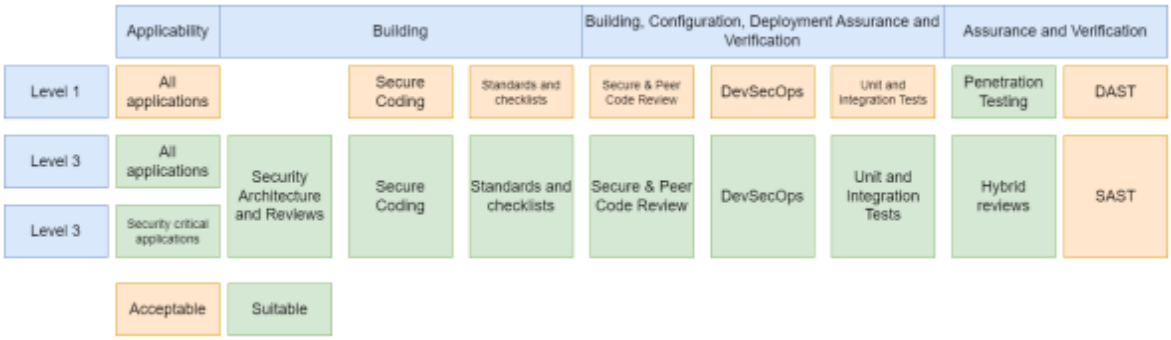


Рисунок 2. OWASP ASVS категорії [45]

Програмний каркас має три рівні перевірки безпеки, які побудовані за принципом послідовного додавання функцій, подібно до CIS Controls. Кожен рівень має свій список вимог безпеки, які можуть бути пов’язані з різними функціями та можливостями програмного забезпечення. [45] Розробник має впровадити ці вимоги у застосунок. Стандарт включає функції, які сприяють більш безпечному створенню, конфігурації та розгортанню застосунків, а також містить процеси перевірки та забезпечення виконання вимог ASVS.

DAST (динамічне тестування безпеки застосунків) та SAST (статичне тестування безпеки застосунків) рекомендуються для перевірки виконання вимог. [45] DAST перевіряє, як застосунок обробляє зловмисні запити в тестовому середовищі, а SAST аналізує код для виявлення вразливостей і недоліків. Ці інструменти зазвичай інтегруються безпосередньо у конвеєр CI/CD. [50]

Рівні перевірки OWASP ASVS:

- Рівень 1 — базовий мінімальний рівень безпеки для застосунків. Він охоплює вразливості, які легко виявити, наприклад, зі списку OWASP Top 10. Для перевірки вимог достатньо лише DAST, оскільки цей рівень повністю тестується методом penetration testing. [45]
- Рівень 2 — рекомендований для більшості вебзастосунків. Цей рівень забезпечує захист від основних ризиків і підходить для застосунків, які обробляють критичні бізнес-функції чи чутливі дані. Перевірка вимог цього рівня потребує доступу до вихідного коду, конфігурацій середовища та документації, оскільки код більше не можна повністю перевірити методом penetration testing. [45]
- Рівень 3 — призначений для найбільш критичних застосунків, які обробляють надзвичайно чутливі дані або використовуються для високозначущих транзакцій. Для цього рівня потрібні значні перевірки безпеки, наприклад, для застосунків у медичній галузі. [45]

Існують також наявні керівництва та чек-листи, які надають детальні інструкції щодо реалізації більшості вимог. [46] Усі вимоги ASVS можуть бути адаптовані під конкретні потреби проєкту, що допомагає зосередитися на найважливіших заходах безпеки для кожного середовища та проєкту. [45] Вимоги у документі ASVS зіставляються зі специфічними категоріями Common Weakness Enumeration (CWE) або деякими вимогами NIST.

1.6. Життєвий цикл розробки програмного забезпечення

Згідно з типовими уявленнями, життєвий цикл розробки програмного забезпечення (SDLC) зазвичай складається з шести основних етапів: планування, визначення, проєктування, створення, тестування та впровадження. Проте з розвитком методології DevOps до SDLC може бути додано сьомий етап — підтримку. Через перетин завдань між етапами планування та визначення їх іноді об'єднують у спільну фазу планування та аналізу вимог. Хоча деякі етапи SDLC можуть мати різні назви у різних підходах, загальний формат залишається однаковим. Головною метою SDLC є створення повного плану для розробки, проєктування, створення та підтримки програмного забезпечення протягом усього його життєвого циклу для досягнення високої якості продукту. [58; 10]

Впровадження правильних заходів безпеки та рекомендацій є критично важливим на всіх етапах життєвого циклу програмного забезпечення. З точки зору організації, вирішення питань, пов’язаних із безпекою, значно дешевше на початкових етапах життєвого циклу програмного забезпечення, оскільки багато проблем у сфері безпеки безпосередньо пов’язані з проєктуванням та архітектурою програмного забезпечення. Крім того, це сприяє підвищенню безпеки програмного забезпечення та полегшує дотримання різних галузевих регуляцій. [20] На наступній схемі (Рисунок 3) наведено основні дії в рамках SDLC.

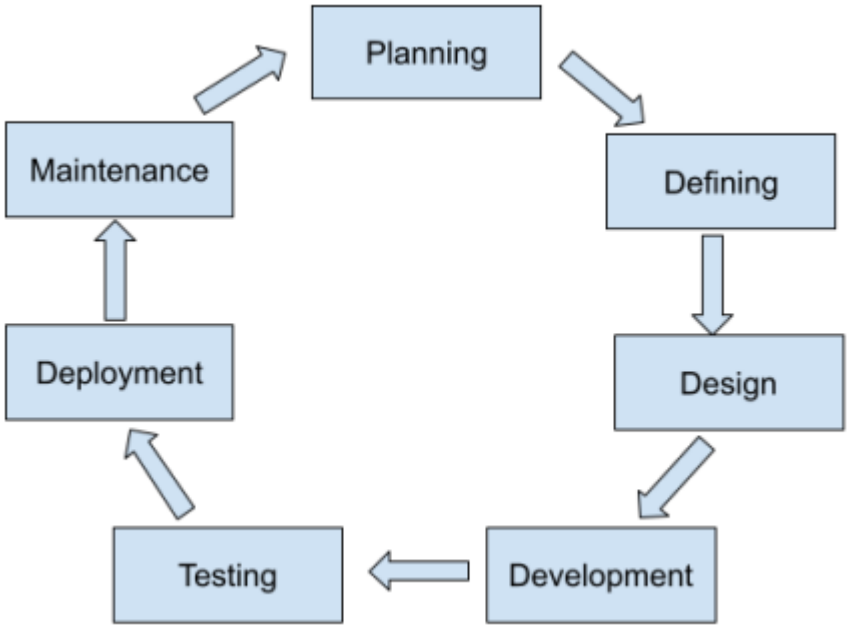


Рисунок 3. Типовий життєвий цикл розробки програмного забезпечення [29]

Першим етапом є фаза планування, під час якої основними завданнями є збір бізнес-вимог клієнта та оцінка здійсненності програмного забезпечення з економічної, технічної та операційної точок зору. Також на цьому етапі планується забезпечення якості та управління ризиками. [58] Після планування зібрані вимоги деталізуються та визначаються на основі аналізу, проведеного у фазі планування. [58] Після завершення цього етапу організація повинна мати специфікацію вимог до програмного забезпечення (SRS) та формальну документацію вимог. Цей документ потім використовується на етапі проєктування. [39]

На цих двох етапах також слід враховувати вимоги безпеки. Вони повинні охоплювати як функціональні аспекти програмного забезпечення, так і інтеграцію різних компонентів. Рекомендується враховувати випадки, коли система може опинитися під атакою. [39]

На основі документа SRS на етапі проєктування розробляється основна архітектура, структура бази даних і програмні інтерфейси. У цій фазі створюється багато різних архітектурних діаграм. Також зазвичай плануються тестові випадки.

Основними заходами, пов'язаними з безпекою на цьому етапі, є ідентифікація та аналіз можливих вразливостей. [39]

Після завершення етапу проєктування починається етап розробки, на якому відбувається фактичне написання коду програмного забезпечення. На цьому етапі реалізуються всі вимоги, визначені в попередніх фазах, тому основні вимоги до безпеки мають бути задокументовані до початку цієї фази. У сучасній розробці тести зазвичай пишуться одночасно з розробкою, тобто модульне тестування є частиною етапу розробки. Якщо проєкт використовує модель розробки, керовану тестами (TDD), тести пишуться до написання основного коду. [41]

Залежно від організації, також є типовим розпочати документування процесу розробки. Ця документація повинна містити всю необхідну інформацію для розробки, розгортання та налаштування програмного забезпечення. Наприклад, вона може включати визначення всіх програмних інтерфейсів (API) та їх використання або документацію структури бази даних програмного забезпечення.

Після завершення розробки програмне забезпечення повинно бути протестоване на етапі тестування. Основна мета цього етапу — перевірити, чи відповідає програмне забезпечення вимогам і чи відсутні помилки або дефекти. Усі знайдені помилки й дефекти виправляються до етапу розгортання. [58] Етап тестування може включати різні типи тестування, такі як модульне й інтеграційне тестування. У модульному тестуванні перевіряється окремий компонент програмного забезпечення, а в інтеграційному тестуванні — взаємодія між різними компонентами. [41] Для модульного та інтеграційного тестування широко використовується автоматизація. Тестування безпеки має включати перевірку моделей атак і загроз. Також рекомендується проводити динамічне тестування безпеки застосунків (DAST) і тестування на проникнення. [39]

Після завершення етапу тестування програмне забезпечення переходить до етапу розгортання, де його впроваджують у виробниче середовище. [58] Етап розгортання зазвичай включає процес прийняття, створення пакета розгортання та розробку плану відновлення. З поширенням хмарних обчислень автоматизовані інсталяції оновлень програмного забезпечення у виробниче середовище стали звичайною практикою, що також є частиною цього етапу.

Останнім і зазвичай найдовшим етапом є етап підтримки, який охоплює весь час між розгортанням програмного забезпечення та завершенням його використання. Це безперервний процес, під час якого виправляються помилки, дефекти та проблеми з безпекою, якщо вони виникають. [26]

Існує багато моделей SDLC: каскадна, ітераційна, спіральний процес, Agile, V-подібна модель та прототипування. [10] Кожна з цих моделей прийняла формальне визначення SDLC.

Існує кілька моделей SDLC, але найбільш визнаними є традиційна каскадна модель, ітераційна та Agile.

Каскадна модель є першою створеною моделлю SDLC і працює за принципом каскаду, що означає послідовний порядок етапів без їхнього накладання. Наступний етап може початися тільки після завершення попереднього. У каскадній моделі використовуються ті ж етапи SDLC, що й у класичному підході. Одним із головних недоліків каскадної моделі є її негнучкість до змін у вимогах, що обмежує її використання проєктами, де всі вимоги відомі на початку. [58] Найбільш підходящими для цієї моделі є проєкти, де контроль якості має вирішальне значення, оскільки в каскадній моделі велика увага приділяється документуванню та плануванню перед початком розробки. [58; 11]

Ітераційна модель працює подібно до каскадної, але після етапу тестування процес повертається до фази планування, і програмне забезпечення

вдосконалюється на основі нових знань і уточнених вимог. У цій моделі не всі вимоги можуть бути відомі на початку проєкту, тому проєкт починається з невеликого набору вимог. [58]

Модель Agile набула популярності завдяки різним Agile-методологіям, таким як Scrum і Kanban. Це ітеративний підхід, у якому різні етапи SDLC повторюються багато разів протягом усього проєкту. Agile слідує структурі SDLC, за винятком того, що на кожній ітерації відсутня фаза підтримки. Це означає, що після кожної ітерації створюється й постачається робоче програмне забезпечення. Коли застосунок готовий і впроваджений у виробниче середовище, фаза підтримки додається до SDLC Agile-моделі. [55]

Сьогодні багато команд розробки інтегрують практики розробки та операцій (DevOps) у фази Agile-моделі. Використання DevOps передбачає співпрацю команд розробки та операцій для покращення процесів розробки. DevOps зазвичай об’єднує етапи розробки, тестування та впровадження в єдиний узгоджений цикл у межах Agile-моделі. [43] Найбільш поширеними практиками є використання автоматизованих інструментів, які сприяють безперервній інтеграції/безперервній доставці (CI/CD) і автоматизованому тестуванню. Інтеграція безпечних процесів DevOps може впливати на організації багатьма способами, такими як автоматизоване тестування, розгортання та процеси моніторингу, що позитивно впливає на безпеку програмного забезпечення. [43] Однак це може суперечити відповідності деяким вимогам безпеки, особливо через відсутність аудитів і перевірок, а також нечіткий контроль доступу до чутливих даних. [43]

Інтеграція безпечної розробки програмного забезпечення у SDLC є ключовою для дотримання галузевих стандартів. Це особливо важливо для розробників у високорегульованих галузях, таких як фінансові технології або медицина, де розуміння безпечних практик розробки є пріоритетом. [12] Багато організацій впроваджують у свої політики обов’язкові навчання для розробників

щодо останніх оновлень безпеки. Такі навчання часто обумовлені необхідністю дотримання загальноприйнятих галузевих стандартів безпеки, наприклад, серії ISO 27000.

Було визначено шість різних практик безпеки, кожна з яких тісно пов’язана із сучасними процесами в рамках Agile-моделей розробки. [12] Проте було виявлено, що лише фахівці з безпеки приділяють цим аспектам увагу у своїй щоденній роботі. Тому безпечні практики повинні бути ефективніше інтегровані в Agile-процеси розробки програмного забезпечення. [12]

2. ДОСЛІДЖЕННЯ СТАНАРТІВ БЕЗПЕКИ ПРОГРАМНОГО КАРКАСУ ЯК БАЗОВОЇ КОНЦЕПЦІЇ

2.1. Стандарти безпеки

Було проведено багато досліджень щодо різних стандартів безпеки програмного забезпечення та їх використання. Одне з досліджень порівнювало популярні галузеві стандарти безпеки для виявлення, які стандарти, такі як Common Criteria та OWASP ASVS, можна застосовувати протягом життєвого циклу розробки застосунків і як це робити. [52] Результати дослідження показали, що окремі стандарти безпеки можуть не охоплювати всі вимоги до безпечної розробки програмного забезпечення. Тому організаціям рекомендується впроваджувати кілька стандартів і рекомендацій, а не покладатися на один стандарт, щоб забезпечити відповідність регуляційним вимогам або сертифікаційним стандартам. [47]

Інше дослідження порівнювало стандарт ISO 27001 з програмним каркасом кібербезпеки NIST (CSF) на високому рівні, щоб визначити переваги та недоліки обох підходів і допомогти організаціям вибирати відповідні каркаси відповідно до їхніх потреб. [64] Було також розглянуто взаємне узгодження інших каркасів і рекомендацій, таких як CIS Top-20. Основний висновок полягав у тому, що CSF і ISO 27001 підходять для різних типів організацій: CSF більше підходить для технологічних компаній, а ISO 27001 — для комерційних через акцент на різних атрибутах або аспектах безпеки. Наприклад, CSF є більш «підходящим для впровадження схем контролю», тоді як ISO 27001 надає точніші та детальніші інструкції. Для досягнення найкращого результату організаціям рекомендується використовувати комбінацію різних стандартів. [64]

У попередніх дослідженнях застосування стандартів на рівні застосунків для оцінки відповідності було досить поширеним. OWASP ASVS виділяється як найпоширеніший стандарт цього рівня. Одне з досліджень зосереджувалося на виявленні повторюваних функцій корпоративних вебзастосунків, які можуть бути

оцінені на відповідність OWASP ASVS. Було рекомендовано, щоб організації дотримувалися вимог, використовуючи контрольний список, і адаптували вимоги OWASP відповідно до своїх потреб. [57] Деякі частини вимог можуть перекриватися, але результати дослідження можуть бути використані як інструкції для ухвалення рішень, оцінки безпеки застосунків, а також допомоги розробникам у таких сферах, як налаштування середовища, проєктування архітектури та вибір технологій. [57]

Досліджень щодо використання контрольних заходів CIS було проведено небагато. Одне з досліджень критично оцінювало контрольні заходи CIS, не аналізуючи технічні аспекти окремих заходів, а скоріше перевіряючи різні твердження, зроблені CIS, та їх обґрунтованість. [33] Було запропоновано ідеї щодо покращення певних недоліків цих заходів, що може бути корисним для створення програмного каркасу. Також зазначалося, що для великих організацій необхідно більше планування та пріоритизації під час проведення оцінки ризиків для застосунків. Було підкреслено, що оцінка ризиків завжди залежить від контексту, і кожен захід безпеки слід адаптувати до конкретного середовища, а не покладатися лише на оцінку ризиків, проведену CIS. [33]

Ці аспекти слід враховувати при проєктуванні артефакту.

2.2. Розробки програмного каркасу

Одне з досліджень розробило автоматизовану модель програмного каркасу для ухвалення рішень, яка може використовувати методи оптимізації для вибору найбільш економічно вигідних критичних заходів безпеки CIS. [16] Модель враховує залишковий ризик, витрати та обмеження зручності використання. Основною метою каркасу було спрощення впровадження всіх заходів безпеки, оскільки інтеграція всіх механізмів може бути «економічно невигідною через обмеження бюджету та бізнес-політики». Модель продемонструвала хороші результати для великих підприємств і може бути покращена за допомогою

алгоритмів машинного навчання. Це свідчить про те, що контрольні заходи CIS можуть бути ефективно використані у великих організаціях для управління ризиками. [16]

Інше дослідження об’єднало OWASP ASVS у структурну та аналітичну модель для оцінки рівнів безпеки вебзастосунків. [62] Це дозволило отримати розуміння сильних і слабких сторін аналізованих застосунків. Оскільки «сирі дані, отримані лише з ASVS, важко обробляти та аналізувати», нова модель була потрібна для підвищення розуміння стану безпеки системи та ухвалення обґрунтованих рішень. За допомогою цієї моделі вимоги OWASP ASVS розглядаються більш детально, що забезпечує комплексний погляд на безпеку системи. [62]

Ще одне дослідження розробило систематичний підхід до зіставлення заходів безпеки OWASP ASVS з рекомендаціями Управління валютного нагляду Сінгапуру (MAS) щодо управління технологічними ризиками та кібергігієни. [59] Це показало, що впровадження ASVS у процес розробки дозволяє компаніям досягати високого рівня відповідності регуляціям. Основною проблемою залишається інтерпретація регуляцій, оскільки вони часто є надто загальними. У фінансовому секторі організації часто стикаються з численними регуляціями, які іноді перетинаються. Таким чином, створення універсального програмного каркасу є ідеальним рішенням. Це підтверджує, що міжнародні стандарти, такі як OWASP ASVS, можуть поєднуватися з галузевими стандартами. [59]

Деякі дослідження також показали, що кілька стандартів безпеки можуть бути об’єднані для створення уніфікованого програмного каркасу. Один із каркасів для управління ризиками кібербезпеки у хмарних обчисленнях був розроблений на основі стандартів, таких як ISO 27001 і NIST CSF. Він складався з 21 кроку, що пропонували чітку дорожню карту для впровадження кібербезпеки. Інший каркас для хмарних IT-систем охоплював весь SDLC застосунку, використовуючи загальноприйняті стандарти інформаційної безпеки. Основною

ідеєю було забезпечення безпеки хмарних систем із самого початку їх розробки, оскільки інтеграція заходів безпеки у вже створені системи може бути складною та дорогою. [60]

Ще один підхід передбачав розробку моделі пріоритизації контрольних заходів безпеки, яка допомагає ідентифікувати найважливіші вразливі елементи на основі різних критеріїв оцінки. Ця модель дозволяє зменшити витрати ресурсів, оптимізувати оцінку заходів контролю та допомагає організаціям вибирати заходи безпеки відповідно до їхніх потреб. [53]

Ці дослідження підтверджують, що стандарти безпеки можуть бути успішно інтегровані в єдиний програмний каркас, який забезпечує ефективне управління ризиками та відповідність вимогам регуляцій. [10]

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ АРХІТЕКТУРИ ПРОГРАМНОГО КАРКАСУ

3.1. Дослідження у галузі проєктування (Design Science Research)

Метою цієї дипломної роботи є розробка програмного каркасу, який може бути використаний як посібник із забезпечення безпеки програмного забезпечення на всіх етапах життєвого циклу розробки. Обраний метод дослідження — це дослідження у галузі проектування (Design Science Research, DSR). [27]

Основна ідея всіх визначень DSR полягає у тому, що це підхід, який базується на артефактах. Артефакти в рамках DSR можуть бути будь-якими створеними об'єктами, у яких дослідницький внесок відображений у їхньому дизайні, наприклад, нові моделі, методи або фізичні конструкції. Розробка артефакту повинна базуватися на існуючих теоріях і знаннях про проблемну область. [30]

Для DSR оцінка результатів дизайну є критично важливою і тісно пов'язана з проектуванням самого артефакту. Оцінка артефакту проводиться не тільки в контексті середовища, але також повинна враховувати його дизайн і внесок у нові знання. Метою DSR є забезпечення того, щоб артефакт робив значний внесок у наукову базу знань. [35; 61]

Один із підходів у DSR об'єднує різні визначення цього методу в єдину стандартну структуру, яку можна використовувати як керівництво для майбутніх досліджень. Ця структура складається з шести основних етапів: [30]

1. Ідентифікація проблеми та мотивація дослідження

2. Визначення цілей для вирішення

3. Проєктування та розробка артефакту

4. Демонстрація артефакту

5. Оцінка артефакту

6. Комунікація

Перша діяльність визначає проблему дослідження та обґрунтовує цінність, яку рішення може принести. На другому етапі описуються більш широкі цілі запланованого артефакту. Після цього артефакт проєктується і створюється з необхідною функціональністю. На четвертому етапі демонструється, як створений артефакт вирішує визначену проблему. Демонстрація може бути реалізована через експерименти, моделювання або навіть кейс-дослідження. Після демонстрації артефакт оцінюється на основі того, наскільки добре він вирішує визначену проблему. Нарешті, весь процес і результати комунікуються зацікавленим сторонам. [30]

Хоча DSRM має зазначені шість етапів, дослідники не обов'язково мають дотримуватися суворого порядку. Початок дослідження може відбуватися з будь-якого з перших чотирьох етапів залежно від контексту дослідження. Ці точки входу важливі, оскільки вони допомагають структурувати весь процес дослідження та оцінити артефакт. Крім того, дослідження у галузі проєктування є ітеративним процесом, що дозволяє вдосконалювати та змінювати кожну діяльність протягом дослідження. [30]

Можливі точки входу в дослідження:

- Ініціація, орієнтована на проблему
- Ініціація, орієнтована на об’єкт
- Ініціація, орієнтована на проєктування та розробку
- Ініціація, орієнтована на клієнта або контекст

Ініціація, орієнтована на проблему, є основним підходом у дослідженнях, який починається з першого етапу на основі спостереження за проблемою. Підхід,

орієнтований на об’єкт, починається з другого етапу і зазвичай базується на потребах галузі або дослідження, де потрібен артефакт. Дослідження, орієнтоване на проєктування та розробку, починається з третього етапу, зосереджуючись на дизайні артефакту та проблемній області. Нарешті, підхід, орієнтований на клієнта або контекст, починається з четвертого етапу, часто базуючись на спостереженні за існуючим практичним рішенням. [30]

Потреба у створенні запланованого програмного каркасу безпеки виникла з потреб компанії Evitec Solutions Oy, тому найбільш відповідною точкою входу є ініціація, орієнтована на об’єкт.

Існує кілька типів артефактів, які підходять для DSR. Нижче наведено шість різних типів артефактів:

Таблиця 1. Різні типи артефактів [30]

Тип	Опис
Model	Спрощене представлення реальності, яке допомагає зрозуміти, як працює концепція.
Framework	Базова концептуальна структура, яка може бути використана для планування та ухвалення рішень.
Algorithm	Набір формальних логічних інструкцій, що описують певний процес.
Instantiation	Фактична апаратна чи програмна реалізація.

Method	Надає практичні інструкції, які можна використовувати для впровадження певного явища.
Construct	Концепція, яка була розроблена на основі набору інших ідей і тверджень.

Серед різних типів артефактів найбільш підходящим для цієї дипломної роботи є програмний каркас (тобто мета-модель). Концепція програмного каркасу також передбачає можливість його налаштування під конкретні потреби, що є важливим для створюваного артефакту. [42]

Для оцінки артефактів у дослідженнях у галузі проєктування доступні різні методи, такі як прототипи, кейс-дослідження, ілюстративні сценарії, логічні аргументи, експертні оцінки або експерименти.[30] З них найбільш підходящим методом для оцінки артефакту цієї дипломної роботи було обрано експертну оцінку. У рамках експертної оцінки один або кілька фахівців у своїй галузі аналізують артефакт на основі своїх знань і досвіду. Це означає, що професіонали оцінюють розроблений програмний каркас, аналізують його відповідність вимогам і виявляють потенційні проблеми. [30]

Оцінка артефакту має проводитися ex-post, тобто після того, як артефакт буде створений.

3.2. Застосування методології дослідження

Демонстрація використання артефакту для вирішення проблеми дослідження в реальному контексті була неможливою. Тому вирішення проблеми буде продемонстровано за допомогою напівструктурованих інтерв’ю, у яких експерти галузі з організації висловлять свою думку щодо програмного каркасу. На основі відповідей, отриманих під час інтерв’ю, буде оцінено придатність

програмного каркасу та його відповідність поставленій проблемі. Також інтерв’ю використовуються для оцінки артефакту.

Оцінка артефакту базується на стандартах, використаних у програмному каркасі, та їх відповідності найпоширенішим галузевим рекомендаціям у Європі. Серед цих рекомендацій — інструкції з інформаційно-комунікаційних технологій (ІКТ) та документація з управління ризиками, опубліковані Європейським банківським управлінням. Отримані знання з інтерв’ю також використовуються для повторного проєктування та вдосконалення створеного каркасу за потреби.

3.2.1. Процес інтерв'ю

Оскільки обраним методом оцінки для дипломної роботи була експертна оцінка, важливо було отримати думки галузевих експертів про програмний каркас. Напівструктуроване інтерв’ю є комбінацією структурованих і неструктурованих інтерв’ю. Питання в такому форматі зазвичай відкриті, що сприяє більш вільним обговоренням. Порядок і кількість питань також можуть варіюватися, і інтерв’юер може задавати додаткові запитання учасникам. [48; 21]

Зазвичай напівструктуровані інтерв’ю тривають від 30 до 60 хвилин. [49] Через специфіку дипломної роботи було обрано 45 хвилин як оптимальний час для збору всіх релевантних даних від учасників. Було рекомендовано записувати інтерв’ю. [44] Усі зустрічі проводилися онлайн через віддаленість. [48]

Хоча оптимальна кількість учасників для інтерв’ю становить від 9 до 17 осіб, через обмежену кількість доступних експертів з організації було проведено інтерв’ю з трьома людьми. [34] Питання в інтерв’ю починалися зі слів «що», «хто», «де», «коли» або «як», відповідно до рекомендацій. [48] Через проведення інтерв’ю різними мовами питання могли не повністю відповідати формулюванням у Таблиці 2, але їхній контекст був збережений.

Немає встановлених правил щодо кількості питань у напівструктурованих інтерв'ю, тому інтерв'юер самостійно обирає правильну кількість. Для цього випадку було визнано, що 8-10 питань є оптимальними.

Таблиця 2: Питання для інтерв'ю

ID	Опис
IQ1	Як часто вам доводиться враховувати вимоги безпеки програмного забезпечення у вашій роботі?
IQ2	Виходячи з вашого досвіду, як здійснювався моніторинг та управління інтеграцією вимог безпеки у проєктах, над якими ви працювали?
IQ3	З якими значними викликами ви стикалися при забезпеченні відповідності стандартам безпеки протягом процесу розробки? Як ви їх вирішували?
IQ4	Чи стикалися ви з ситуаціями, коли стандартизований програмний каркас безпеки, подібний до створеного, міг би бути корисним у минулих проєктах? Якщо так, як?
IQ5	Яка ваша думка щодо зручності використання та практичності впровадження

	запропонованого програмного каркасу безпеки?
IQ6	Як ви вважаєте, як запропонований програмний каркас безпеки покращує безпеку програмного забезпечення та відповідність регуляторним вимогам організації?
IQ7	Наскільки обрані стандарти безпеки відповідають потребам фінансової індустрії?
IQ8	Які основні переваги інтеграції контрольних заходів CIS та OWASP ASVS у SDLC організації?
IQ9	Чи є якісь аспекти каркасу, які потребують подальшого вдосконалення, і які ваші пропозиції щодо підвищення його ефективності та зручності?

Оскільки розроблений програмний каркас мав бути інтегрований у SDLC компанії, для отримання найбільш релевантних відповідей від учасників інтерв'ю було важливо, щоб учасниками були ті, хто враховує вимоги безпеки програмного забезпечення на всіх етапах життєвого циклу. Це обмежило вибір учасників до проєктних менеджерів або системних архітекторів, які зазвичай беруть участь у всіх фазах розробки програмного забезпечення — від збору вимог до його підтримки.

Оскільки метою було інтегрувати каркас у загальний процес розробки та оцінити його відповідність вимогам безпеки у фінансовій сфері, для інтерв'ю були обрані три архітектори програмного забезпечення з компанії. Архітектори могли надати більш детальні думки про реальне використання каркасу, ніж проєктні менеджери, оскільки вони мають більше практичного досвіду у фазі розробки. Крім того, вони є ключовими учасниками процесів проєктування архітектури та впровадження програмного забезпечення. [12]

Питання для інтерв'ю були поділені на дві секції: вступні питання та цільові питання. Це допомогло учасникам замислитися над поточними практиками безпеки у своїх проєктах та висловити ідеї щодо застосовності розробленого каркасу. Варто зазначити, що всі відповіді базувалися на особистому досвіді з каркасом, а також на припущеннях, оскільки реальна інтеграція каркасу в проєкт не могла бути здійснена.

Для аналізу результатів інтерв'ю буде використано дедуктивний тематичний аналіз. У контексті дипломної роботи тематичний аналіз означає, що результати інтерв'ю будуть аналізуватися з метою пошуку спільностей та закономірностей між відповідями, з урахуванням наявної теорії. [24]

3.3. Вимоги до проєктування

На основі огляду літератури були визначені деякі загальні характеристики, які використовувалися при проєктуванні артефакту. Було відзначено, що формат контрольного списку є стандартним для картографування програмних каркасів. OWASP ASVS широко використовується для оцінювання та в різних налаштованих картографуваннях, що дозволяє ефективно аналізувати безпеку застосунків.

Раніше проведені дослідження показали, що впровадження різних стандартів є дуже специфічним для контексту, що означає, що не всі рекомендації та механізми безпеки OWASP ASVS та CIS Controls можуть бути необхідними. У

процесі проєктування слід враховувати потреби організації. Також можливі перекриття між різними регуляціями та рекомендаціями, тому вимоги повинні оцінюватися під час проєктування.

Проєктований програмний каркас має бути широким і не обмежувальним, щоб його можна було розширювати в майбутньому. Це також одна з ключових вимог організацій. Дослідження показали, що впровадження програмного каркасу як частини SDLC компанії є ідеальним рішенням, і реалізація його на початкових етапах розробки програмного забезпечення зменшує витрати на інтеграцію в процес розробки. Загалом, ідея налаштованого програмного каркасу полягає в тому, щоб оцінювати та впроваджувати механізми безпеки більш ефективно в рамках організації. Такі програмні каркаси зазвичай працюють як посібники або інструкції для розробників, які допомагають дотримуватися регуляцій.

Розробка артефакту базувалася як на попередніх наукових дослідженнях, так і на вимогах організації. Вимоги організації до артефакту наведені у Таблиці 3.

ID	Опис
REQ1	Програмний каркас має бути інтегрований у SDLC організації.
REQ2	Програмний каркас має бути придатним до оновлення у майбутньому.
REQ3	Програмний каркас має використовувати відповідний формат.
REQ4	Програмний каркас має інтегрувати вимоги OWASP ASVS v4.0.3.

REQ5	Програмний каркас має інтегрувати заходи безпеки CIS Controls v8.
REQ6	Програмний каркас має дозволяти картографування без перевантажень. Подібні вимоги мають бути об'єднані.
REQ7	Програмний каркас має пропонувати відповідні інструкції для повсякденної роботи.
REQ8	Програмний каркас має бути достатньо точним, щоб персонал знав, які вимоги слід враховувати і на якому етапі.
REQ9	Програмний каркас має забезпечувати базову лінію для безпечної розробки застосунків.
REQ10	Програмний каркас має відповідати загальним вимогам фінтех безпеки.
REQ11	Програмний каркас має полегшувати управління вимогами безпеки в проєктах.
REQ12	Програмний каркас не повинен перешкоджати поточній діяльності проєктів.
REQ13	Програмний каркас має забезпечувати можливості для моніторингу управління каркасом.

Під час розробки програмного каркасу для початкового етапу було вирішено використовувати найбільш детальний рівень картографування для обох стандартів. Це дозволяє чіткіше визначати прогалини між стандартами. Використання цього методу також виявляє проблеми під час інтеграції організаційних та рівня застосунків вимог.

У вимогах до програмного каркасу було враховано галузь, у якій працює організація, а також її поточні процеси розробки.

3.4. Проєктування

На основі вимог організації та рекомендацій з літератури для програмного каркасу було обрано формат таблиці Excel. Процес картографування вимог та механізмів безпеки у SDLC включав наступні кроки:

1. Аналіз кожної категорії ASVS окремо та виявлення спільних рис із механізмами безпеки CIS.
2. Аналіз різних фаз SDLC та їх основної мети з точки зору організації.
3. Картографування кожного механізму безпеки CIS та вимоги ASVS до фаз SDLC організації.
4. Картографування кожної вимоги ASVS до відповідного механізму безпеки CIS.

У процесі розробки враховувалася SDLC організації. SDLC організації містить типові шість фаз життєвого циклу програмного забезпечення. Фаза "Вимоги" в SDLC організації об'єднує діяльності фаз "Планування" та "Визначення", згаданих раніше.

На основі аналізу механізмів CIS Controls і вимог OWASP ASVS було вирішено додати нову фазу — Фаза 0: Загальні вимоги. Основною метою цієї фази є включення вимог, які не корелюють безпосередньо з типовими фазами

SDLC та повинні бути враховані на організаційному рівні або до початку проєкту. Ця фаза в основному складається з механізмів безпеки CIS Controls v8.

SDLC організації спроєктована так, що кожна фаза має одну загальну ціль для аспектів безпеки програмного забезпечення. Ця структура була адаптована для використання в фінансовому секторі ЄС і має спільні риси для всіх етапів. Основною вимогою SDLC є те, що рівень OWASP ASVS 2 є базовим стандартом для всіх розробок застосунків.

Безпекові практики організації та їхнє розташування в різних фазах SDLC були враховані під час розробки програмного каркасу.

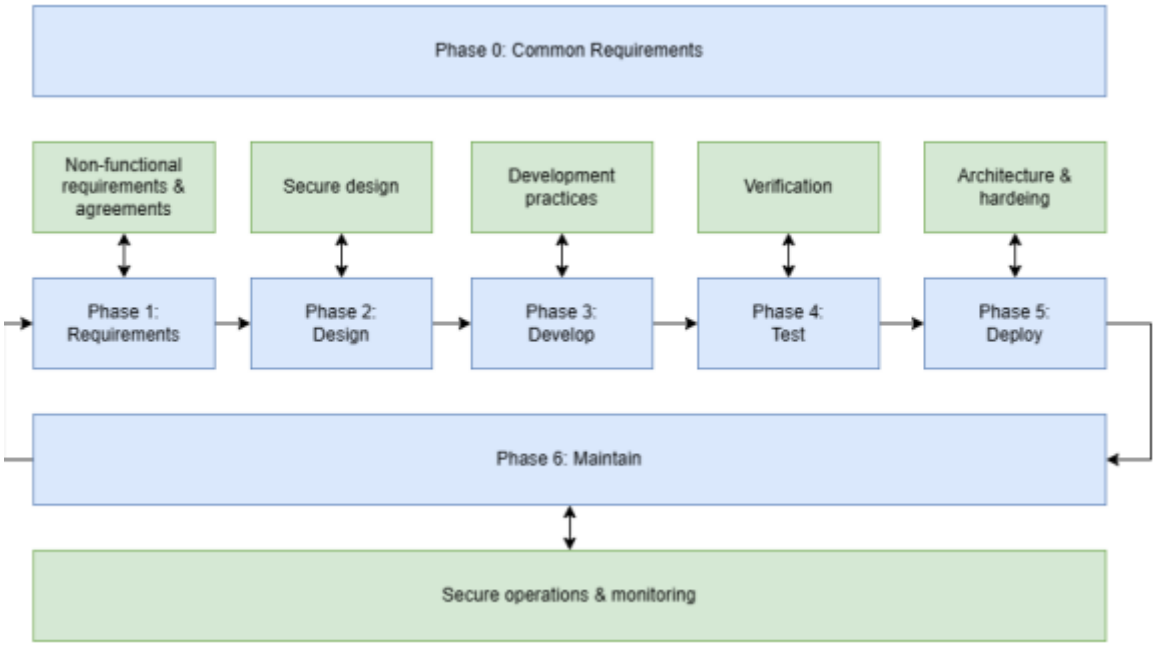


Рисунок 4. Життєвий цикл розробки програмного забезпечення (SDLC) організації

Під час проєктування було зроблено кілька ключових спостережень, які вплинули на розробку програмного каркасу. Картографування лише механізмів CIS і вимог ASVS разом не забезпечує достатніх інструкцій для ефективного застосування рекомендацій до застосунку та середовища. Щоб спростити процес впровадження, кожен стандарт повинен бути пов’язаний із загальною структурою SDLC конкретної організації. Це дозволить розробникам і архітекторам знати, які

механізми безпеки та рекомендації слід враховувати на кожному етапі розробки застосунку.

CIS Controls більше зосереджені на організаційних заходах безпеки для різних аспектів підприємства, тоді як OWASP ASVS спрямовані на забезпечення безпеки вебзастосунків. Тому найбільш оптимальним використанням програмного каркасу є застосування CIS Controls для загального рівня безпеки організації, а рівнів OWASP ASVS — для безпеки вебзастосунків. Також було помічено, що багато вимог ASVS охоплюють кілька різних механізмів CIS.

Було виявлено, що механізми CIS, які стосуються застосунків, здебільшого пов’язані з посиленням архітектури середовища та практиками безпечної розробки, залишаючи багато на інтерпретацію розробників. Таким чином, інтеграція двох стандартів є ключовою. Жоден із стандартів не пропонує простих методів впровадження, тому завдання визначення способів їх виконання лягає на впроваджувача. Це також зазначалося в літературі. Хоча це не було вирішено під час розробки, рекомендується врахувати це у майбутньому.

Під час розробки було зазначено, що не всі рекомендації ASVS застосовуються до всіх підконтролів основних рівнів CIS. Також майже кожна вимога ASVS у певних аспектах охоплює механізми CIS 16.1 і 16.10. Таким чином, врахування всіх інших вимог ASVS забезпечує відповідність згаданим підконтролям.

Деякі підконтролі CIS можна пов’язати з безпекою на рівні застосунків, але вони представлені як загальні заходи безпеки на рівні компанії. Картографування всіх контролів CIS до основної структури SDLC виявилося складним, оскільки багато підконтролів повинні виконуватися на організаційному рівні й не пов’язані з життєвим циклом програмного забезпечення. Таким чином, була додана фаза Фаза 0: Загальні вимоги до SDLC, що дозволяє кожен контроль CIS віднести до однієї з фаз SDLC.

Було також зазначено, що багато вимог ASVS і механізмів CIS можуть бути частиною кількох фаз SDLC залежно від їхньої інтерпретації. Тому їхнє остаточне розташування у каркасі базувалося на контексті організації, для якої був створений програмний каркас, і на її процесах.

Розроблений каркас містить дев'ять стовпців:

Таблиця 4: Колонки каркасу

Назва колонки	Опис колонки
Number	Відповідний номер у документах CIS Controls v8 або OWASP ASVS v4.0.3.
Description	Опис вимоги або механізму безпеки з документації.
Level 1	Найнижчий рівень безпеки для застосунку.
Level 2	Базові стандарти для всіх програмних забезпечень організації.
Level 3	Тільки для найбільш критичних з точки зору безпеки застосунків.
Addressing ASVS Requirements	Якщо деякі вимоги OWASP ASVS безпосередньо відповідають механізмам CIS, врахування вимог ASVS у цьому полі автоматично враховує і відповідний механізм CIS.
Type	Тип вимоги.

Additional notes for implementation	Детальніші коментарі щодо вимоги або механізму безпеки, які можуть вплинути на процес її впровадження. Наприклад, посилання на OWASP Top 10 можуть спростити процес впровадження. OWASP Top 10 і вимоги ASVS мають подібні CWE, і попередні знання Top 10 можуть полегшити впровадження вимог ASVS.
Implemented	Повинно бути "YES" або "NO" залежно від того, чи була впроваджена вимога.
How requirement has been addressed	Уточніть, як саме була виконана вимога.

Рівневі колонки в програмному каркасі були налаштовані та не повністю відповідають документації CIS Control v8 або OWASP ASVS. У документації OWASP ASVS рівні корелюють із критичністю безпеки застосунків; натомість рівні (так звані групи впровадження IG) у CIS більше орієнтовані на організаційні та інфраструктурні механізми безпеки.

Для забезпечення зручності використання програмного каркасу було вирішено об’єднати групи впровадження CIS і рівні OWASP ASVS у єдині рівні. Це означає, що групи впровадження CIS (IG) безпосередньо відповідають рівням OWASP ASVS у цьому каркасі.

Остаточні колонки каркасу показані на Рисунках 5 та 6 нижче:

Number	Description	Level 1	Level 2	Level 3	Addressing ASVS Requirement
SDLC Phase 0: Common requirements					
CIS Controls					
1.1 Establish and Maintain detailed enterprise asset invertory		X	X	X	

Рисунок 5. Завершені заголовки програмного каркасу, частина 1

Type	Additional notes for implementation	Implemented	How requirement has been addressed

Рисунок 6. Завершені заголовки програмного каркасу, частина 2

Вимоги були класифіковані за різними типами на основі їхніх схожостей. У Таблиці 5 наведено детальні пояснення для кожної з 16 категорій:

Таблиця 5. Основні категорії програмного каркасу

Категорія	Опис
Access control	Вимоги, пов’язані з контролем доступу. Включає автентифікацію, авторизацію, управління сесіями, роботу з токенами доступу та аутентифікаторами. Наприклад, OWASP ASVS 2.10.2: Перевірте, чи є облікові дані не стандартним обліковим записом.
Business logic	Вимоги, пов’язані з бізнес-логікою. Наприклад, OWASP ASVS 11.1.5: Перевірте, чи застосунок має бізнес-логічні обмеження або валідацію для захисту від ризиків чи загроз, визначених за допомогою моделювання загроз.
Configuration	Вимоги, пов’язані з налаштуванням середовища, а також із вимогами до розгортання. Наприклад, CIS 16.8: Розділіть виробничі та невиробничі системи.
Cryptography	Вимоги, пов’язані з шифруванням та алгоритмами. Включає управління секретами, паролями та зв’язком. Наприклад, OWASP ASVS 6.3.1: Перевірте, чи

	створюються випадкові значення за допомогою схваленого криптографічного модуля.
Data management	Вимоги, пов’язані з управлінням даними. Наприклад, CIS 11.3: Захистіть дані відновлення.
Error handling	Вимоги, пов’язані з обробкою помилок. Наприклад, OWASP ASVS 7.4.3: Перевірте, чи визначено обробник помилок "останнього засобу", який обробляє всі необроблені виключення.
File management	Вимоги, пов’язані з управлінням файлами. Наприклад, OWASP ASVS 1.12.1: Перевірте, чи зберігаються файли, завантажені користувачами, за межами кореневої папки вебсайту.
Input validation	Вимоги до валідації введення користувачем, а також валідації даних HTTP-запитів. Наприклад, OWASP ASVS 2.1.2: Перевірте, чи дозволяються паролі довжиною 64 символи або більше.
Logging	Вимоги, пов’язані з журналюванням. Наприклад, OWASP ASVS 7.1.4: Перевірте, чи кожна подія журналу включає

	необхідну інформацію для детального розслідування хронології події.
Monitoring	Вимоги до моніторингу окремих аспектів програмного забезпечення. Включає вимоги до мереж, сповіщень і управління патчами. Наприклад, CIS 13.9: Розгорніть управління доступом на рівні портів.
Organizational	Організаційні вимоги, які слід застосовувати на організаційному рівні. Наприклад, CIS 4.5: Реалізуйте та управляйте брандмауером на пристроях кінцевих користувачів.
Privacy	Вимоги, пов’язані із захистом даних, такими як GDPR, PII та обробка конфіденційної інформації. Наприклад, OWASP ASVS 8.1.1: Перевірте, чи захищає застосунок конфіденційні дані від кешування в компонентах сервера.
Project management	Загальні вимоги до проєктів, наприклад, управління вразливостями. Наприклад, OWASP ASVS v1.1.3: Перевірте, чи всі user stories містять функціональні обмеження безпеки.

Secure practices	Загальні практики безпечної розробки. Наприклад, OWASP ASVS v14.2.5: Перевірте, чи підтримується каталог інвентаризації для всіх сторонніх бібліотек, що використовуються.
Testing	Вимоги до тестування. Наприклад, CIS 16.13: Проведіть тестування на проникнення застосунку (penetration testing).
Training	Вимоги до навчання персоналу. Наприклад, CIS 14.4: Навчіть працівників найкращим практикам роботи з даними.

Для відображення різних концепцій у програмному каркасі використовувалася кольорова диференціація. На Рисунку 7 нижче показано, як використовувалося кольорове маркування:

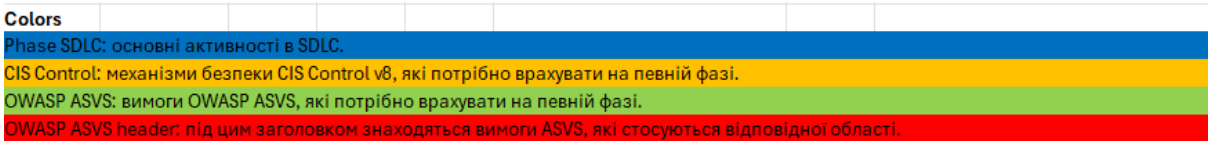


Рисунок 7. Концепції програмного каркасу

Картографування вимог і механізмів безпеки виконувалося на найнижчому можливому рівні, тобто кожен підконтроль CIS та вимога OWASP ASVS безпосередньо прив'язувалися до певної фази SDLC. Через велику кількість вимог було вирішено розміщувати кожну вимогу OWASP ASVS у її власному розділі, що зробило каркас більш зручним для читання.

Це рішення було прийняте для полегшення диференціації різних вимог ASVS, оскільки багато з них можуть бути складними та впливати на різні аспекти програмного забезпечення без більш широкого контексту. У підконтролях CIS схожих елементів не виявлено; тому вони розміщені під заголовком CIS Controls.

Посилання на стандарти використовують налаштований формат, який відрізняється від рекомендацій, зазначених у документації OWASP. Формат посилань на вимоги OWASP ASVS виглядає як v<версія>-<ідентифікатор_вимоги>, щоб розрізняти кожну версію. Проте для потреб організації номер версії не був потрібний, тому використовувався лише ідентифікатор вимоги.

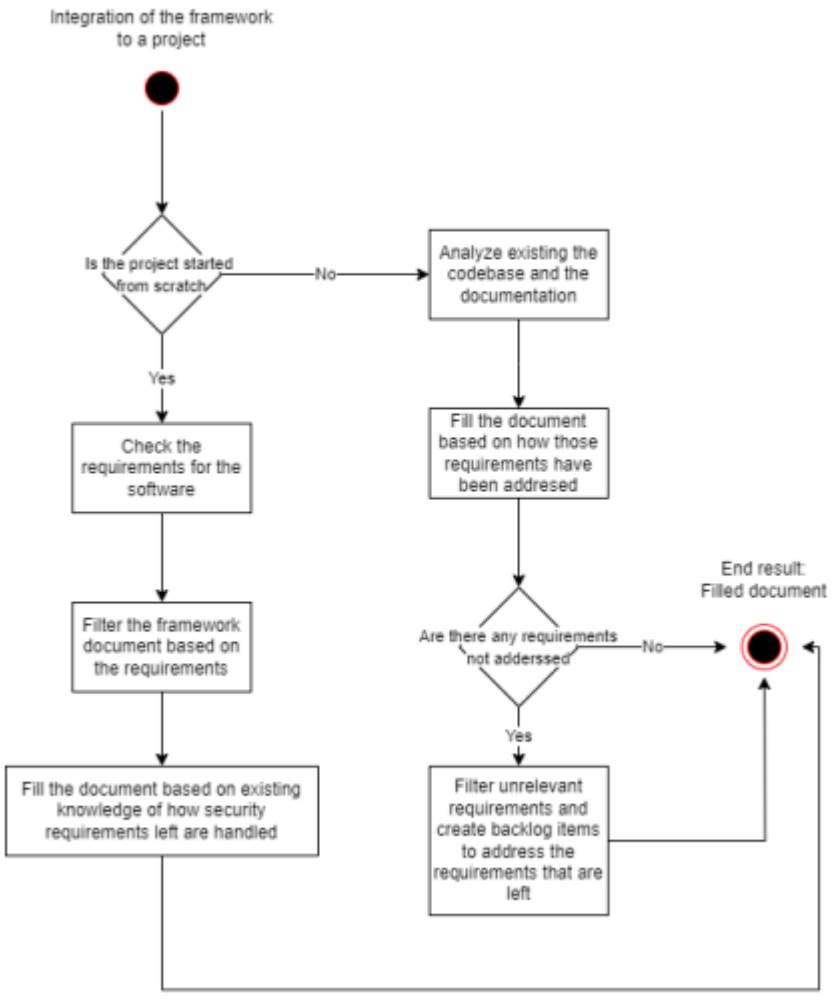


Рисунок 8. Початкова інтеграція програмного каркасу до проєкту

На Рисунку 8 вище показані етапи початкової інтеграції програмного каркасу. Загальний процес полягає в тому, що персонал, відповідальний за основні рішення щодо безпеки проєкту, заповнює документ на основі свого розуміння контексту проєкту. Процес буде однаковим як для існуючих, так і для нових проєктів. Єдина відмінність полягає в тому, що для існуючих проєктів документ заповнюється з посиланням на наявну документацію та вихідний код. Для нових проєктів посилання базуються на наявних знаннях та вимогах конкретного програмного забезпечення.

Рекомендується відфільтрувати непотрібні вимоги з документа, оскільки в програмному каркасі є багато вимог, специфічних для певних функцій. Основні випадки використання програмного каркасу для кожної зацікавленої сторони наведено в Таблиці 6 нижче.

Таблиця 6. Зацікавлені сторони та їхнє використання програмного каркасу

Зацікавлена сторона	Використання
Розробник	• Звернення до програмного каркасу під час розробки. • Покращення знань про практики безпечної розробки.
Архітектор/Спеціаліст з безпеки	• Звернення до програмного каркасу під час розробки. • Заповнення початкового каркасу на основі знань про проєкт.

	• Додавання нових вимог до каркасу за потреби. • Перевірка документа при додаванні нової функції, чи є відповідні вимоги для виконання цієї функції.
Проєктний менеджер	• Відстеження відповідності вимогам безпеки програмного каркасу.

Випадки використання програмного каркасу залежать від ролі у проєкті. Коли розробник починає працювати над проєктом, каркас буде спрямовувати його на дотримання практик безпечної розробки. Розробник також може звертатися до документації каркасу під час розробки за необхідності.

Проєктний менеджер здійснює моніторинг того, щоб вимоги безпеки були враховані.

Ключовим користувачем програмного каркасу є спеціаліст з безпеки або провідний архітектор проєкту. Вони заповнюють початкову документацію каркасу, а також додають нові вимоги до документації. Ці нові вимоги можуть бути пов’язані з безпековими запитами клієнта або новими кращими практиками безпеки. Крім того, вони перевіряють, чи є в каркасі вимоги безпеки, які потрібно виконати для нової функції.

3.5. Приклад використання програмного каркасу

У цьому розділі розглядається приклад використання програмного каркасу в умовах штучного середовища. Рисунки 9 і 10 заповнені відповідною інформацією для конкретного механізму CIS Control і вимоги OWASP ASVS у цьому штучному середовищі, і, таким чином, не належать до жодного конкретного проєкту.

У реальному випадку процес заповнення документації буде відповідати шляхові, показаному на Рисунку 8 у Розділі 4.2.

Number	Description	Level 1	Level 2	Level 3	Addressing ASVS Requirements
SDLC Phase 3: Development					
OWASP ASVS					
14.5	Validate HTTP Request Header Requirements				
14.5.1	Verify that the application server only accepts the HTTP methods in use by the application or API, including pre-flight OPTIONS.	X	X	X	-
14.5.2	Verify that the supplied Origin header is not used for authentication or access control decisions, as the Origin header can easily be changed by an attacker.	X	X	X	-
14.5.3	Verify that the cross-domain resource sharing (CORS) Access-Control-Allow-Origin header uses a strict white-list of trusted domains to match against and does not support the "null" origin.	X	X	X	-
14.5.4	Verify that HTTP headers added by a trusted proxy or SSO devices, such as a bearer token, are authenticated by the application.		X	X	-
SDLC Phase 4: Test					
CIS Controls					
11.5	Test Data Recovery		X	X	8.1.5
16.12	Implement Code-Level Security Checks		X	X	10.1.#
16.13	Conduct Application Penetration Testing			X	11.1.#

Рисунок 9. Приклад використання програмного каркасу, частина 1

Type	Additional notes for implementation	Implemented	How requirement has been addressed
Configuration		Yes	Backend has been configured to use following methods and header: Access-Control-Allow-Methods: GET, POST, PUT, DELETE Access-Control-Allow-Headers: Content-Type, Authorization
Access Control	OWASP TOP 10: A07 Identification and Authentication Failures	Yes	Origin header is not used for authentication and backend has been configured to use Access-Control-Allow-Origin: https://exampleSite.com
Access Control	OWASP TOP 10: A07 Identification and Authentication Failures	Yes	See 14.5.2
Access Control	OWASP TOP 10: A07 Identification and Authentication Failures	No	There was no need for this in this project. More details can be seen from https://projectDocument.com
Testing	Test that data recovering works frequently	Yes	Data recovering is tested every month
Secure Practices	Utilize static (e.g., SonarCloud on pipeline) and dynamic analysis tools. Although IG 3 on CIS controls, code-level security checks should be used in Level 2 applications as well. Check for more details in organizations SDLC Test section (SAST chapter)	Yes	SonarCloud has been implemented to the release pipeline.
Testing	More details under CIS 18.1	No	Penetration testing is not required for this project. This was done in accordance to the Project Agreement document

Рисунок 10. Приклад використання програмного каркасу, частина 2

Рисунки 9 і 10 є прикладами використання програмного каркасу в реальному контексті. Інформація в колонках Implemented та How requirement was addressed пропонує приклади того, як каркас може виглядати в реальному проєкті. Рисунки також сильно відфільтровані, щоб показати лише конкретні приклади.

З Рисунків 9 і 10 видно, що OWASP ASVS 14.5 Validate HTTP Request Header Requirements класифіковані для етапу розробки SDLC. Це означає, що основні активності, пов'язані з цими вимогами, повинні виконуватися на цьому етапі. Колонка Type допомагає розробнику зрозуміти тип вимоги. У цьому прикладі більшість вимог OWASP ASVS класифіковані як пов'язані з Access Control, що стосується автентифікації та HTTP-запитів. Класифікація була виконана за допомогою керівництва, зазначеного раніше в Таблиці 5. Колонка Type також використовувалася для фільтрації лише тих вимог, які є частиною конфігурації, контролю доступу, тестування або безпечних практик. Через велику кількість результатів показані лише вибрані приклади.

У цьому прикладі деякі вимоги мають додаткові примітки для впровадження, що допомагають у реалізації цих вимог. У вимогах OWASP ASVS є примітка, що три вимоги входять до OWASP TOP 10. У цьому контексті розробник знає, що для додаткової інформації щодо реалізації потрібно звертатися до документації OWASP TOP 10: A07 Identification and Authentication Failures. Документація OWASP TOP 10 пропонує прямі інструкції для виконання цієї вимоги, яких бракує в основній документації OWASP ASVS.

Для CIS Control 16.13 Conduct Application Penetration Testing є примітка для розробника перевірити рядок CIS 18.1 для отримання додаткової інформації про тестування на проникнення. Таким чином, інформацію для схожих вимог можна знайти в одному місці.

На Рисунку 9 також є червоний ‘X’ у колонці Level 2 для 16.12 Implement Code-Level Security Checks. Це вказує, що практики організації відрізняються від стандарту. У цьому випадку проєкт рівня 2 повинен також впроваджувати перевірки на рівні коду, хоча CIS рекомендує це тільки для рівня 3.

У колонці Implemented видно, що більшість вимог виконані. У колонці How requirement was addressed для тих вимог, які не були виконані, зазначено причину, чому вимога не була виконана. У цьому прикладі проєкт відповідає вимогам рівня 1, тому вимоги інших рівнів не потрібно виконувати. Якщо б проєкт виконувався відповідно до вимог рівня 2 або 3, ці вимоги повинні були б бути виконані, або ж мала б бути чітка причина, чому їх не виконано. Наприклад, для CIS 16.13 Conduct Application Penetration Testing: "Тестування на проникнення не потрібне для цього проєкту. Для отримання додаткової інформації зверніться до документа Project Agreement".

Для інших вимог надано відповідні пояснення, як саме вони були виконані. У цьому прикладі вимоги OWASP ASVS 14.5.2 і 14.5.3 виконані аналогічно. У рядках CIS Controls колонка Addressing ASVS requirement містить вимоги OWASP ASVS, які відповідають конкретному механізму CIS. Для 16.12 Implement Code-Level Security Checks виконання цього механізму потребує повного впровадження всіх вимог OWASP ASVS 10.1 Code Integrity Controls.

3.6. Результати інтерв'ю з галузевими експертами

Демонстрація артефакту була проведена за допомогою експертної оцінки. Експерти галузі проаналізували зміст програмного каркасу, а для збору даних про його ефективність було використано напівструктуроване інтерв’ю. Через конфіденційність імена учасників не розкриваються в дипломній роботі, і з результатів неможливо відрізнити відповіді окремих учасників. Загалом, сумарний досвід трьох опитаних становив майже 70 років роботи в галузі. Відібрані експерти брали участь у численних програмних проєктах протягом своєї

кар'єри, і кожен з них не був частиною одного й того ж проєкту. Посадовою інструкцією кожного з учасників була роль архітектора програмного забезпечення. Учасники були обрані для інтерв'ю через їхнє розуміння розробки програмного забезпечення у фінансовій галузі, значний професійний досвід і загальний обсяг посадових обов'язків.

Інтерв'ю проводилися онлайн, а розроблений каркас був наданий учасникам перед інтерв'ю, щоб вони могли проаналізувати його зміст. Таким чином, учасники мали час ознайомитися з каркасом і підготуватися до інтерв'ю. Учасникам було надано опис програмного каркасу перед інтерв'ю, а також деякі приклади сценаріїв його використання. Це дозволило учасникам заздалегідь ознайомитися з використанням каркасу. Аналогічно, запитання інтерв'ю також були надіслані учасникам за кілька днів до зустрічі, щоб вони могли обміркувати їх.

Тема дипломної роботи, стандарти, використані в каркасі, та сам каркас були пояснені на початку інтерв'ю. Інтерв'ю проводилися з урахуванням того, що учасники мали попереднє уявлення про основні концепції каркасу та про те, як вимоги безпеки реалізуються протягом життєвого циклу програмного забезпечення. Отже, результати інтерв'ю базуються на знаннях учасників у галузі, їхньому досвіді управління безпекою та моніторингу у попередніх програмних проєктах, а також на інформації, наданій автором дипломної роботи до та під час інтерв'ю.

Оскільки інтеграція каркасу в існуючі проєкти не проводилася, оцінка каркасу здійснювалася гіпотетично з використанням прикладів реальних сценаріїв його використання. Основні питання, поставлені під час інтерв'ю, слугували основою для обговорення та визначення основної галузі, в якій каркас міг би використовуватися. Направляючі запитання відкривали дискусію про поточні практики управління безпекою та про те, як інтеграція каркасу могла б

функціонувати в цих умовах. Потім ці теми обговорювалися більш детально за допомогою цільових запитань, що стосувалися самого каркасу.

3.6.1. Направляючі запитання

1. *Як часто вам доводиться думати про вимоги безпеки програмного забезпечення у вашій роботі?*

Аспекти безпеки програмного забезпечення впливають на процеси ухвалення рішень на різних етапах проєкту. Вони найбільш актуальні під час архітектурного планування та інтеграції систем. Зазвичай, питання безпеки базуються на попередніх реалізаціях, коли додаються нові функції до існуючих застосунків. Однак вимоги безпеки є критично важливими під час проєктування нових сценаріїв використання або проєктів. Як правило, існує визначений список вимог, яких необхідно дотримуватися.

Учасники зазначили, що для відстеження аспектів безпеки програмного забезпечення під час проєктування нових сценаріїв можна використовувати чек-листи. Однак виконання вимог безпеки здійснюється тоді, коли вони виникають, і має бути систематично проаналізовано на різних етапах життєвого циклу проєкту.

Вимоги безпеки зазвичай виникають під час продажів. Вимоги OWASP є основними вимогами безпеки, які обговорюють під час обговорення вимог до програмного проєкту, і їх необхідно дотримуватися. Крім того, у фінансовій галузі компанії часто мають власні вимоги безпеки, які організації також повинні враховувати, навіть якщо вони виходять за межі будь-яких стандартів безпеки.

2. *З вашого досвіду, як у проєктах, над якими ви працювали, здійснювався моніторинг та управління інтеграцією вимог безпеки?*

Моніторинг та управління вимогами безпеки зазвичай є більш реактивними, ніж проактивними. Таким чином, вимоги безпеки, як правило, вирішуються та документуються наприкінці проєкту, а не на кожному етапі. Вимоги часто управляються за допомогою таблиць, і документ перевіряється та оцінюється на початкових етапах проєкту або коли від клієнтів надходять нові нефункціональні вимоги.

Вимоги зазвичай є списком умов, які організація повинна виконати певним чином. Однак клієнти зазвичай висувають вимоги, пов’язані із безпекою програмного забезпечення, які необхідно моніторити. Спільною темою було те, що вимоги безпеки в галузі є надзвичайно високими. Таким чином, організації та команди проєктів повинні враховувати більше аспектів безпеки, ніж в інших галузях.

Моніторинг вимог безпеки між різними проєктами може бути складним та ресурсозатратним. Навіть якщо вимоги можна налаштувати, вони зазвичай відповідають деяким вимогам OWASP ASVS, але не можуть бути відображені один до одного. Крім того, під час розробки програмного забезпечення необхідно враховувати регулювання ЄС.

Клієнти також повинні розуміти, як різні вимоги безпеки керуються протягом життєвого циклу проєкту. Клієнти можуть проводити аудити, щоб перевірити, чи правильно інтегровані та керовані вимоги безпеки. Усі учасники також зазначили, що немає систематичного підходу до моніторингу вимог безпеки в проєктах.

3. *Які основні виклики ви стикалися у забезпеченні відповідності стандартам безпеки протягом процесу розробки? Як ви їх вирішували?*

Основні виклики, з якими стикалися учасники, були пов’язані з існуючими проєктами. Підтримка старих застосунків у відповідності до нових галузевих стандартів стає складною, оскільки можуть бути відсутні рекомендації щодо їх

впровадження для старих технологій. Це особливо актуально, оскільки вимоги до безпеки програм постійно змінюються, зростають та визначаються більш детально. Таким чином, виконання нових вимог безпеки та підсилення захисту старих проєктів потребує значних зусиль та часу.

Однак для нових проєктів багато уваги приділяється визначенню кращих практик для впровадження нових вимог безпеки, і реалізація цих вимог залишається на розсуд команди розробників. На даний момент немає стандартного способу виконання вимог безпеки на різних етапах життєвого циклу програмного забезпечення.

Загалом, найскладніші аспекти забезпечення відповідності вимогам безпеки пов’язані з мережею, автентифікацією та авторизацією програмного забезпечення. Особливі виклики виникають під час переходу з локальних рішень до хмарних або SaaS-застосунків. Один із учасників зазначив, що вони вже використовували OWASP Top 10 для тренування своїх розробників.

4. *Чи були у вас ситуації, коли стандартизований каркас безпеки, як запропонований, міг би бути корисним у попередніх проєктах? Якщо так, то як?*

Найкориснішою ситуацією для подібного каркасу в минулому був процес продажів. Учасники обговорювали, що каркас міг би використовуватися як список довідкових матеріалів для потенційних клієнтів, щоб показати, як ці вимоги вже виконані у наявному проєкті.

Приклад: "Ми могли б надати заздалегідь заповнені специфікації клієнту та запитати, чи достатньо цього для їхніх вимог безпеки. Якщо чогось бракує, є пояснення, як цей ризик було ідентифіковано та вирішено".

Крім того, каркас міг би допомогти визначити, як специфічні вимоги клієнтів щодо безпеки були виконані в попередніх проєктах або застосунках організації.

Загалом, єдиний уніфікований каркас безпеки міг би стати гарною базою для розробки будь-якого застосунку та використовуватися як внутрішньо, так і зовнішньо під час процесу продажів. Учасники зазначили, що подібний каркас у минулому міг би бути використаний як матеріал для тренування нових співробітників.

Дотримання каркасу дозволяє оптимізувати вимоги безпеки, зосереджуючись на визначених аспектах, залишаючи місце для врахування специфічних потреб клієнта. На рівні компанії організаційні вимоги, переважно механізми CIS, повинні бути представлені у форматі, що пояснює, як вони вже враховані.

Один із учасників зазначив, що галузеві вимоги клієнтів часто узгоджуються з вимогами OWASP ASVS. Це переважно великі компанії у галузі, які мають жорсткіші вимоги до безпеки. У таких випадках клієнту надається список того, що було враховано, а що ні. Також учасник зазначив, що клієнтська база у країнах Північної Європи лише останні п'ять років почала значно наполягати на дотриманні специфічних вимог безпеки.

3.6.2. Цільові запитання

1. *Що ви думаєте про зручність і практичність впровадження запропонованого каркасу безпеки?*

Загалом, розподіл вимог був відзначений як чудовий спосіб допомогти зосередитися на найбільш критичних вимогах безпеки для певної фази проєкту. Один із учасників зазначив, що в реальному контексті використання каркасу було б більш ітеративним, тобто його можна було б адаптувати під час проєкту, якщо з'ясується, що деякі вимоги та захисні механізми в каркасі є необов’язковими або відносяться до невідповідної фази.

Усі учасники також зазначили, що організація повинна частково заповнити каркас до його впровадження у проєкті. Учасники виявили, що більшість функцій каркасу, особливо механізми CIS, необхідно враховувати на рівні компанії. Механізми CIS у Фазі 0 слід попередньо заповнити перед інтеграцією каркасу в реальний проєкт.

Учасники знову зазначили, що найбільшу цінність каркас може принести під час процесу продажів. Загалом, вони вважали, що мапінг вимог OWASP ASVS до механізмів CIS допомагає виконати вимоги CIS, оскільки визначення механізмів CIS є досить широким. Також мати один уніфікований каркас для всієї організації допомагає керувати стандартними вимогами безпеки у всіх проєктах. Каркас пропонує чудовий спосіб показати, як вимоги безпеки обробляються в організації.

Поточний зміст каркасу був визнаний обґрунтованим для типового розробника, щоб він міг виконувати щоденну роботу. Однак усі учасники зазначили, що деталізація каркасу є прийнятною для загальних вказівок на організаційному рівні. Було запропоновано, щоб розробники використовували OWASP Top 10, а більш деталізований каркас залишався для менеджерів і архітекторів. Усі учасники вважали, що каркас буде використовуватись не щоденно, а переглядатися періодично після кожної фази SDLC. Однією із пропозицій було спочатку використовувати каркас у ширшому контексті програмного продукту, а потім використовувати вузький список у процесі розробки.

Усі учасники зазначили, що каркас є гарним орієнтиром для розробників, оскільки мати стандартний список вимог по всій організації краще, ніж припускати, що розробник враховує питання інформаційної безпеки з пам'яті під час розробки.

Каркас можна було б використовувати для перехресних посилань між різними проєктами, оскільки в інших проєктах вже могли бути способи виконання тих самих вимог безпеки. Це було б корисним, особливо якщо технологічні стеки в проєктах подібні. Однак також обговорювалося, наскільки каркас спрямовує процес розробки, якщо його копіюють із попередніх проєктів. Виконання всіх вимог у каркасі може бути складним у кожному проєкті, оскільки багато вимог у CIS і OWASP залежать від конкретного проєкту та іноді залишаються на відповідальності клієнта. Однак, якщо проєкт є програмним забезпеченням як послуга (SaaS), відповідальність за виконання вимог в основному лежить на організації.

2. *Як, на вашу думку, запропонований каркас безпеки покращує безпеку програмних продуктів і відповідність організації регуляторним вимогам?*

Основне покращення безпеки на організаційному та проєктному рівнях полягає у створенні однакової базової основи для всіх програмних проєктів, що базується на відомих галузевих стандартах. Це гарантує, що правильні вимоги до безпеки будуть враховані. Крім того, інтеграція з SDLC допоможе забезпечити вирішення відповідних питань безпеки на ранніх етапах розробки.

Безпека програмних продуктів покращиться як побічний ефект від покращення знань розробників у сфері безпеки, оскільки використання каркасу самими розробниками сприятиме вирішенню вимог безпеки під час розробки. Загалом, безпекова обізнаність компанії також покращиться. Учасники також зазначили, що дотримання OWASP ASVS може підвищити відповідність регуляторним вимогам, оскільки цей стандарт часто згадується клієнтами під час обговорень. Тому буде легше пояснити, як ці ризики були пом'якшені за допомогою каркасу.

Чек-лист допоможе перевірити, чи все було враховано та запам’ятано. Це особливо помітно, коли функціональність переважає над безпекою програмного

забезпечення у щоденних завданнях. Учасники також зазначили, що для існуючих продуктів заповнення таблиці каркасу надає уявлення про відповідність поточного застосунку вимогам і області для покращення. Порівняння заповнених таблиць каркасу між проєктами ще більше зміцнить безпеку програмного забезпечення організації. Це також сприяє впровадженню найкращих практик у всій організації.

Хоча учасники зазначили, що інтеграція каркасу у новий продукт буде найпростішою, потенційна інтеграція в існуючий проєкт також можлива. Однак слід враховувати існуючі програмні продукти, якщо продукт є стандартизованим або спеціалізованим для клієнта. За словами одного з учасників, інтеграція каркасу буде ідеальною у випадку, коли "залишилося достатньо життєвого циклу (приблизно половина), продукт буде продаватися новим клієнтам, і до проєкту приєднаються нові розробники. Тоді такий каркас буде ідеальним, особливо якщо раніше для цього проєкту/продукту нічого суперечливого не було зроблено."

3. Наскільки придатні вибрані стандарти безпеки для фінансової індустрії?

Усі учасники раніше використовували деякі програмні каркаси OWASP у своїх проєктах. Загальна думка щодо OWASP ASVS була такою, що він пропонує фундаментальні аспекти для розробки застосунків. Учасники відзначили його практичність та детальні інструкції щодо вирішення різних питань безпеки в різних технологіях. Крім того, вони зазначили його відповідність вимогам індустрії, оскільки багато компаній фінансового сектору вже його впроваджують. Один з учасників запропонував зробити цей стандарт обов’язковим для всіх нових програмних продуктів.

Знання про CIS Controls були мінімальними, але на основі короткої інформації, наданої перед і під час інтерв’ю, було зазначено, що деякі механізми можуть бути не найважливішими для компанії, що працює у фінансовому секторі,

проте вони є розумною базою для загальної інформаційної безпеки. Конкретні механізми не були названі.

Рівень 2 каркасу був визнаний відповідним для задоволення вимог галузі. Один із учасників зазначив, що у випадку проєктів типу SaaS, рівень 2 може бути недостатньо суворим для виконання вимог галузі, і в таких проєктах слід враховувати деякі вимоги рівня 3.

4. *Які основні переваги інтеграції CIS Controls та OWASP ASVS у SDLC організації?*

Кожен учасник підкреслив організаційну вигоду від уніфікованого каркасу, який консолідує важливу інформацію про безпеку програмного забезпечення, замість того, щоб розділяти її між окремими джерелами. Вони також виділили перевагу спільного бачення процедур і підходів в організації. Це може стати культурним аспектом всередині компанії, що допоможе в майбутніх проєктах завдяки наявності стандартних практик безпеки на всіх етапах розробки. Таким чином, персонал зможе переходити з проєкту в проєкт без необхідності вивчати нові підходи до безпеки.

Ще однією суттєвою перевагою є демонстрація відповідності конкретним стандартам безпеки та вимогам під час організаційних оцінювань. Дотримання галузевих стандартів підвищує потенціал залучення нових клієнтів.

5. *Чи є аспекти каркасу, які потребують подальшого вдосконалення, і які у вас є пропозиції для покращення його ефективності та зручності використання?*

Усі учасники зазначили, що фінансова індустрія має суворі регуляторні вимоги, яких повинні дотримуватись організації, а клієнти зазвичай висувають свої вимоги до безпеки. Тому розроблений каркас повинен бути розширюваним у майбутньому. Ці доповнення можуть базуватися не лише на існуючих стандартах, але й на кращих практиках, яких слід дотримуватись у SDLC загалом. Наприклад,

інтеграція відповідних вимог конкретного клієнта в каркас або врахування фінського законодавства.

Також були висловлені занепокоєння щодо того, як враховувати нові версії стандартів та зміни у вже існуючих розділах. Наприклад: «Як ми змінюємо каркас на основі нових оновлень, і водночас як це впливає на застосунки, розроблені за попередніми версіями каркасу?»

Крім того, як випливало з попередніх питань, існувала загальна думка, що найбільш розумним поділом було б мати загальний каркас на рівні компанії, як і було запропоновано, але водночас мати вузький список вимог, призначений для використання розробниками у їхній роботі. Розробнику не потрібно було б безпосередньо враховувати конкретні вимоги CIS або OWASP під час розробки. Приклади того, що саме необхідно виконати в кожному механізмі безпеки, особливо для CIS, могли б допомогти краще зрозуміти конкретний механізм.

Також повинна бути можливість зв'язати різні способи впровадження вимог із вже існуючою документацією, через що двоє учасників висловили свої зауваження щодо обраного формату електронної таблиці. Як альтернатива, міг би бути запропонований інший спосіб інтеграції каркасу для реального використання у проєкті.

3.7. Оцінка програмного каркасу за вимогами

Було неодноразово зазначено під час інтерв’ю, що обрані стандарти для каркасу пропонують відповідні рекомендації, які відповідають загальним вимогам фінтех-індустрії. Проте більшість їхнього аналізу базувалась на знаннях стандартів OWASP ASVS. Тому потрібна додаткова оцінка, щоб повністю визначити, чи виконує REQ10 вимогу про відповідність каркасу загальним вимогам безпеки фінтех-індустрії.

З цієї причини оцінка REQ10 для вимог CIS і OWASP ASVS була співставлена з інструкціями Європейського банківського управління (EBA) щодо інформаційно-комунікаційних технологій (ICT) та рекомендаціями з управління ризиками у сфері безпеки. Ці інструкції є обов'язковими для виконання кожною компанією у фінтех-індустрії Європи в їхніх організаційних процесах та проєктах. [22]

У наступній таблиці 7 наведено основні розділи інструкцій EBA та їх використання в оцінці: [22]

Розділ	Використання
3.1 Пропорційність	Пропущено
3.2 Управління та стратегія 3.2.1 Управління 3.2.2 Стратегія 3.2.3 Використання сторонніх постачальників	Використано Використано Пропущено Пропущено
3.3 Каркас управління ризиками ICT та безпеки 3.3.1 Організація та цілі 3.3.2 Ідентифікація функцій, процесів і активів 3.3.3 Класифікація та оцінка ризиків 3.3.4 Пом'якшення ризиків 3.3.5 Звітність 3.3.6 Аудит	Пропущено Пропущено Пропущено Пропущено Пропущено Пропущено Пропущено Пропущено
3.4 Інформаційна безпека 3.4.1 Політика інформаційної безпеки	Використано Використано Використано

3.4.2 Логічна безпека 3.4.3 Фізична безпека 3.4.4 Операційна безпека ІСТ 3.4.5 Моніторинг безпеки 3.4.6 Огляд, оцінка та тестування інформаційної безпеки 3.4.7 Навчання та підвищення обізнаності з інформаційної безпеки	Пропущено Використано Використано Використано Використано
3.5 Управління операціями ІСТ 3.5.1 Управління інцидентами та проблемами ІСТ	Використано Використано
3.6 Управління проєктами та змінами ІСТ 3.6.1 Управління проєктами ІСТ 3.6.2 Придбання та розробка систем ІСТ 3.6.3 Управління змінами ІСТ	Використано Пропущено Використано Пропущено
3.7 Управління безперервністю бізнесу 3.7.1 Аналіз впливу на бізнес 3.7.2 Планування безперервності бізнесу 3.7.3 Реагування та відновлення 3.7.4 Тестування планів 3.7.5 Кризові комунікації	Використано Пропущено Пропущено Використано Пропущено Пропущено
3.8 Управління відносинами з користувачами платіжних послуг	Пропущено

Декілька розділів були повністю пропущені в процесі оцінки, оскільки вони не містили жодних вимог, пов’язаних із жодним зі стандартів, використаних у програмному каркасі, і, таким чином, не є частиною таблиці оцінки. Таким чином, найважливіші рекомендації для оцінки каркасу наведені в Розділі 3.4, Інформаційна безпека. Більш детальне зіставлення стандартів, використаних у каркасі, з рекомендаціями EBA наведено в (Додатку 1).

Список вимог у (Додатку 1) є стислою версією вимог - рекомендацій EBA «Керівні принципи управління ризиками ICT і безпеки».

На основі оцінки зіставлення стандартів, використаних у каркасі, і рекомендацій щодо ICT безпеки, згаданих у рекомендаціях EBA, більшість рекомендацій можна виконати, задовольняючи одну або кілька вимог стандартів каркасу. Результати вказують, що CIS Controls переважно відповідають рекомендаціям EBA (див. докладніші результати у Додатку 1). Однак, як зазначено в Таблиці 7, зіставлення було виконано лише для вимог, пов’язаних із заходами безпеки, згаданими у стандартах. Це означає, що багато вимог рекомендацій EBA було пропущено. Загалом, CIS Controls тісно узгоджуються з рекомендаціями EBA щодо інформаційної безпеки. Оскільки рекомендації EBA є всеосяжними та орієнтовані більше на загальну організаційну безпеку, не дивно, що не багато вимог OWASP ASVS можна безпосередньо зіставити з будь-якими рекомендаціями. Хоча OWASP ASVS не напряму співвідносяться з рекомендаціями EBA, інтерв’ю свідчать, що вимоги OWASP ASVS є відповідними та узгоджуються зі специфічними потребами європейської фінтех-індустрії. У таблиці 8 показано, як артефакт виконав вимоги, зазначені в Таблиці 3.

ID	Опис	Як вимогу виконано	Виконано (Так/Ні/Частково)
REQ1	Програмний каркас має бути	Програмний каркас було поділено	Так

	інтегрований у ЖЦРЗ (SDLC) організації.	на основні сім фаз ЖЦРЗ організації, а вимоги OWASP ASVS та CIS розподілено по фазах SDLC.	
REQ2	Програмний каркас має бути оновлюваним у майбутньому.	Додавання нових вимог до програмного каркасу можливе, якщо вони можуть бути віднесені до однієї з фаз ЖЦРЗ.	Частково
REQ3	Програмний каркас має використовувати підхожий формат	Використовується формат таблиці, що полегшує відстеження вимог.	Частково
REQ4	Програмний каркас має інтегрувати вимоги OWASP ASVS v4.0.3.	Усі вимоги OWASP ASVS v4.0.3 були розподілені по фазах ЖЦРЗ.	Так
REQ5	Програмний каркас має інтегрувати вимоги CIS Controls v8.	Усі вимоги CIS Controls v8 були розподілені по фазах ЖЦРЗ.	Так
REQ6	Програмний каркас має об’єднувати різні стандарти без дублювання.	Вимоги OWASP ASVS були зіставлені з CIS Controls, що дозволяє виконувати ASVS вимоги для	Так

	Подібні вимоги мають бути зведені в один контроль.	відповідності CIS safeguards.	
REQ7	Програмний каркас має надавати зручні рекомендації для щоденної роботи.	Додано колонку типу, що полегшує категоризацію різних вимог.	Так
REQ8	Програмний каркас має бути достатньо точним, щоб персонал знав, які вимоги враховувати і на якій фазі.	Виконано через додавання колонки «Додаткові примітки для реалізації», яка допомагає персоналу зрозуміти специфіку вимоги.	Частково
REQ9	Програмний каркас має забезпечити базу для розробки захищених додатків.	Інтерв'ю та попередні дослідження показали, що обрані стандарти забезпечують надійну основу для безпечної розробки.	Так
REQ10	Програмний каркас має відповідати загальним фінансово-технологічним вимогам безпеки.	Каркас узгоджується з фінансово-технологічними вимогами, викладеними Європейським	Так

		банківським управлінням (EBA).	
REQ11	Програмний каркас має спростити управління вимогами безпеки в проєктах.	Консолідація всіх вимог безпеки у таблицю значно полегшує управління.	Так
REQ12	Програмний каркас не повинен заважати поточній діяльності проєктів.	Інтерв'ю не виявили перешкод у поточних проєктах, але реальна оцінка потребує додаткового впровадження.	Частково
REQ13	Програмний каркас повинен забезпечувати можливості моніторингу керування програмним каркасом.	Є колонки для перевірки реалізації та опису способу виконання вимог.	Так

На основі фінальної оцінки програмного каркасу було виявлено кілька спільних спостережень. Розроблений каркас переважно задовольняє вимоги, визначені організацією. Жодна з вимог не залишилася повністю невиконаною, хоча деякі не змогли бути повністю оцінені.

Для більш детальної оцінки REQ2 необхідно впровадити каркас у один або декілька реальних проєктів. Тому ця вимога не може бути класифікована як повністю виконана, оскільки інтеграція каркасу в поточний проєкт не була

здійснена. Водночас, за відгуками учасників, додавання нових вимог до каркасу є цілком досяжним завданням.

Щодо REQ3, загальний формат каркасу визнано відповідним у контексті цієї роботи. Однак остаточний формат може змінитися під час його повної інтеграції в SDLC організації. Це питання було порушено під час інтерв'ю, хоча повна інтеграція каркасу в SDLC організації не входила до меж цього дослідження.

Варто також зазначити, що розроблений у рамках цього дослідження каркас є базовим контрольним списком для впровадження та відстеження OWASP ASVS і CIS Controls у проєктах. Для повного задоволення REQ8 та REQ12 необхідно оцінити практичне використання каркасу в реальних умовах. Це дозволило б отримати глибше уявлення про його ефективність та визначити будь-які додаткові вдосконалення, необхідні для його комплексного впровадження.

3.8. Обговорення

Під час розробки програмного каркасу стало очевидним, що деякі заходи CIS орієнтовані на організаційний рівень, але також можуть бути застосовані на рівні додатків. Наприклад, CIS Control 5 Account Management і 6 Access Control Management надають заходи для управління обліковими записами та доступом на рівні організації, однак ці аспекти також слід враховувати при створенні середовищ для додатків. Це особливо важливо при налаштуванні хмарних середовищ. Тому, якщо організації використовують ці загальні організаційні стандарти, такі як CIS, їм слід враховувати контекст. Вказівки CIS настільки широкі, що іноді важко розрізнити вимоги, які мають виконуватися організацією, та ті, які повинні виконуватися в рамках окремих проєктів. Саме тому в SDLC було додано Фазу 0. [59]

Використання каркасу в процесі продажу згадувалося багато разів під час інтерв'ю. Хоча основним призначенням каркасу було регулювання, управління та інтеграція безпечних практик у програмне забезпечення, попередньо заповнений документ каркасу може бути корисним і в контексті продажів. Проте використання каркасу у процесі продажу може вимагати певних змін залежно від потреб бізнес-кейсу. Під час інтерв’ю OWASP Top 10 згадувалося настільки часто, що було вирішено додати посилання на цей документ у каркас для спрощення процесу впровадження. [36] Однак слід зазначити, що OWASP Top 10 не задовольняє всі вимоги Рівня 2 і має використовуватися лише як додаткова документація, оскільки Top 10 та ASVS можуть бути зіставлені з одними й тими ж загрозами безпеці CWE. Організація OWASP навіть зазначає, що Top 10 є переважно інформаційним документом і не має використовуватися як стандарт.

Інтерв’ю показали, що звичайні розробники можуть не бажати вивчати або дотримуватися великого списку вимог безпеки. Однак дотримання стандарту чи подібного каркасу, як описано у цій роботі, має інтегрувати аспекти безпеки в повсякденну роботу розробників, а не залишати ці аспекти лише на розгляд

фахівців із безпеки чи архітекторів. [12] Тому для організацій вигідно дотримуватись деяких галузевих стандартів безпеки та максимально інтегрувати їх у процес розробки програмного забезпечення.

Однак, крім наявності керівних вказівок, працівникам необхідно мати достатні знання про аспекти безпеки, інакше виконання вимог стане складним завданням. Саме тому навчання відіграє ключову роль у розумінні співробітниками принципів безпечної розробки програмного забезпечення. Відповіді на інтерв'ю показують, що деякі команди вже впровадили навчання з питань безпеки у свої проєкти. Основним висновком є те, що каркас можна використовувати як навчальний матеріал для нових співробітників. Зазвичай такі навчання передбачені багатьма галузевими стандартами безпеки, такими як ISO 27000 та CIS Controls. Подібні вимоги також містяться в керівництві EBA. У каркасі є 11 контрольних заходів і вимог, які певною мірою стосуються навчання персоналу. [12]

Одним із цікавих аспектів літературного огляду та інтерв'ю була слабка присутність CIS Controls. Учасники здебільшого аналізували каркас на основі своїх знань про OWASP ASVS, а оцінка CIS Controls була мінімальною. Цей розрив у знаннях між двома стандартами був присутній і в літературному огляді, причому більшість попередніх досліджень зосереджувались на вимогах OWASP. [33] Тому для ефективної оцінки CIS Controls було використано додатковий метод, у якому вони зіставлялися з поширеними галузевими стандартами фінансових технологій, визначеними Європейським банківським органом (EBA). Обмежене зіставлення, проведене в цій роботі, показало, що CIS Controls певною мірою відповідають більшості вимог інформаційної безпеки, встановлених EBA.

Також, оскільки вимоги в документації EBA є широкими, організація повинна самостійно інтерпретувати їх відповідно до своїх потреб. Це зробило зіставлення програмного каркасу з керівними принципами досить складним завданням, оскільки рекомендації в CIS також є доволі загальними. Таким чином,

відповідність керівним принципам залежить від інтерпретації захисних заходів CIS. Проте результати цього дослідження показують, що стандарти, використані в програмному каркасі, відповідають багатьом вимогам, встановленим EBA.

З інтерв'ю стало зрозуміло, що деякі проєкти використовують процеси DevOps. Однак, якщо організації дотримуються рекомендацій програмного каркасу, деякі з цих проблем можуть бути мінімізовані, оскільки багато контрольних заходів спеціально орієнтовані на правильне налаштування та захист додатків і середовищ. Хоча їхнє дослідження зосереджувалося на Agile-моделі, результати цього дослідження вказують, що інтеграція безпечних практик є бажаною для будь-якої моделі розробки програмного забезпечення. [43]

На основі оцінки розроблений програмний каркас пропонує чіткі способи для організацій керувати безпечними процесами розробки в різних підходах до розробки програмного забезпечення. Інтеграція програмного каркасу в практики розробки програмного забезпечення в межах організації призведе до загального покращення цих процесів. Крім того, як побічний продукт, загальні знання про безпеку в організації також повинні покращитися, а відповідність вимогам програмного каркасу потребуватиме регулярного навчання розробників. Як було зазначено в інтерв'ю, програмний каркас може навіть позитивно вплинути на процес продажу. Дотримання стандартів у каркасі також допомагає організації частково відповідати галузевим нормам. Однак для повної відповідності організації не можуть покладатися лише на два стандарти, використані в каркасі; тому рекомендується дотримуватися інших, більш загальних стандартів, таких як ISO 27000, для виконання всіх вимог. Як зазначалося в інтерв’ю, вимоги Рівня 2 можуть бути недостатніми для проєктів SaaS. Тому організаціям слід проводити аналіз ризиків у таких випадках і додавати більше вимог з Рівня 3 для цих проєктів. [12]

Програмний каркас також допомагає командам розробників систематично враховувати вимоги безпеки, що, як показали інтерв’ю, було необхідністю.

Аналогічно, дотримання каркасу на кожному етапі проєкту розробки програмного забезпечення зменшує навантаження під час зовнішніх аудитів. Одним із викликів, виявлених під час інтерв’ю, було те, що клієнти хотіли знати, як управляються вимоги безпеки протягом усього життєвого циклу проєкту. [51] Результати інтерв’ю також вказують, що використання каркасу пропонує відповідні способи впоратися з цим завданням. Застосування каркасу сприяє моніторингу впровадження вимог безпеки на кожному етапі життєвого циклу програмного забезпечення, що дозволяє проєкту бути краще підготовленим до зовнішніх аудитів, оскільки каркас демонструє аудиторам, як виконано аспекти безпеки.

Як показали результати інтерв'ю, гнучкий формат каркасу може допомогти в процесі впровадження вимог. Можна створювати попередньо заповнені електронні таблиці для різних проєктів, які використовують схожі технології, що можна використовувати як посилання під час розробки. Таким чином, аналогічні підходи можуть застосовуватись для дотримання однакових вимог у таких випадках. Крім того, під час інтерв'ю багато разів згадувалося, що один із головних переваг каркасу – це аспект документації. Таким чином, усі вимоги до безпеки та їх відповідність зібрані в одному місці, а не в різних джерелах. Це стає корисною документацією для внутрішнього та зовнішнього використання. Таким чином, каркас може бути використаний як основа для демонстрації того, як організація наразі враховує або буде враховувати різні вимоги безпеки клієнтів і чи є ці вимоги достатніми для клієнта.

Результати також свідчать, що каркас допомагає команді зосереджуватись на найбільш критичних вимогах безпеки для кожного етапу розробки. Знахідки також вказують, що спільний програмний каркас в межах організації спрощує перехід розробників між проєктами.

3.9. Обмеження та майбутні дослідження

Дослідження має певні обмеження, які впливають на результати роботи. По-перше, артефакт був оцінений у мінімальному обсязі, тому результати не можуть бути узагальнені для ширшого контексту. Відповідність стандартів різним етапам SDLC (життєвого циклу розробки програмного забезпечення) також виконувалась у контексті процесів і практик розробки конкретної організації. Це означає, що зміст програмного каркасу є специфічним для цієї організації, і в інших організаціях результати можуть відрізнятися.

Каркас був продемонстрований виключно через його оцінку експертами організації, тобто він аналізувався тільки на основі їхніх знань у цій галузі. Для оцінки залучили лише трьох експертів, що обмежує точність результатів. Оптимальна кількість учасників для якісних інтерв'ю становить від 9 до 17 осіб. [34] Проте через обмежену кількість і доступність галузевих експертів у дослідженні взяли участь лише троє.

Крім того, всі мапування були виконані автором дипломної роботи на основі власних знань предметної області. Інструкції EBA та контрольні пункти CIS зазвичай мають досить загальні визначення своїх вимог, тому їх контекстуалізація залежить від впроваджувача. Це означає, що мапування, виконане різними дослідниками, може відрізнятися від того, що зробив автор.

Каркас оцінювався за рекомендаціями, встановленими EBA, але для оцінки використовувалися лише найважливіші вимоги, пов’язані з каркасом. Це означає, що каркас не було зіставлено з повним переліком вимог EBA, оскільки більшість із них не стосуються безпеки інформаційно-комунікаційних технологій (ICT). Однак це відкриває можливість для подальшого аналізу, як організації можуть відповідати різним індустріальним вимогам у майбутньому. Наприклад, наскільки ширші організаційні стандарти, такі як серія ISO 27000 або різні стандарти NIST, відповідають вимогам, встановленим різними банківськими асоціаціями по

всьому світу, такими як ЕВА або Федеральна резервна система США. Деякі вимоги не змогли бути віднесені до виконаних через обмеження методу демонстрації, використаного у роботі. Як зазначено раніше, слід провести реальну імплементацію каркасу та оцінити її в майбутньому, щоб визначити, чи ці вимоги виконуються.

На основі результатів оцінки оновлення застарілого програмного забезпечення також виявилося проблемою, яка вимагає значних ресурсів. Каркас, як показано, не пропонує чітких покрокових інструкцій для вирішення таких проблем. Це варто розглянути в майбутньому, оскільки в організаціях по всьому світу, ймовірно, існують десятки тисяч подібних ситуацій, і це може стати темою для подальших досліджень.

Штучний інтелект (ШІ) та хмарні обчислення стають дедалі більш поширеними в сучасному технологічному ландшафті. Їхній аналіз був поза межами цього дослідження, але вони стануть більш актуальними в майбутньому, особливо в суворо регульованих галузях, таких як фінтех. Ось чому рекомендується додавати інші стандарти та рекомендації до програмного каркасу. Відсутність заходів безпеки, пов'язаних із ШІ, ймовірно, обумовлена тим, що використовувані версії стандартів були створені кілька років тому, і популярність ШІ-рішень зросла лише нещодавно. Дослідження в цій сфері у майбутньому може бути цікавим, оскільки ці технології розвиваються, а стандарти безпеки змінюються.

ВИСНОВКИ

Цей розділ коротко висвітлює результати дослідження та їх відповідність дослідницьким запитанням. Основною метою дослідження було розробити програмний каркас безпеки, який можна інтегрувати безпосередньо в життєвий цикл розробки програмного забезпечення (SDLC) організацій. Дослідницькі

запитання роботи стосувалися можливості поєднання двох різних стандартів безпеки інформаційних технологій у єдиний програмний каркас, який можна використовувати для моніторингу практик безпечної розробки та надання рекомендацій щодо безпечної розробки програмного забезпечення на кожному етапі його життєвого циклу. Крім того, метою роботи було визначення потенційних переваг впровадження такого індивідуального каркасу в організаціях. Вона також намагалася встановити, чи достатньо стандарти, включені в каркас, відповідають галузевим вимогам безпеки. Програмний каркас був розроблений на основі практик, згаданих в існуючій літературі, і вимог, встановлених організацією.

Результатом дослідження став програмний каркас безпеки, який інтегрує OWASP ASVS і контрольні пункти CIS до SDLC. Каркас був продемонстрований через напівструктуровані інтерв'ю з архітекторами організації, які висловили свою думку щодо його життєздатності. Фактична інтеграція до проєктів не проводилася, тому висновки є гіпотетичними, але базуються на думках галузевих експертів щодо можливої інтеграції каркасу в реальні проєкти. Загальна оцінка успішності каркасу була проведена на основі результатів інтерв'ю та відповідності керівним принципам EBA, зазначеним у (детальніше у Додатку 1). На основі оцінки в Таблиці 8 каркас задовольнив більшість вимог. Деякі вимоги не можуть бути позначені як виконані через обмеження роботи, тому для належної оцінки інтеграції каркасу до реального проєкту необхідно провести реальне впровадження.

Основними висновками роботи є те, що контрольні пункти CIS та вимоги OWASP ASVS можна інтегрувати в SDLC організацій. Це забезпечує рекомендації щодо інтеграції індивідуальних каркасів у життєвий цикл розробки програмного забезпечення. На основі демонстрації та оцінки були визначені кілька ключових переваг, які інтеграція програмного каркасу безпеки може принести організаціям. Серед основних переваг: покращення моніторингу виконання вимог безпеки, покращення процесів продажу, полегшення інтеграції

нових розробників у проєкт, підвищення знань про практики безпечної розробки програмного забезпечення та загальне покращення безпеки програмних продуктів організації. Крім того, узгодження вимог ASVS із ширшими контрольними заходами CIS допомагає забезпечити відповідність вимогам. Результати також свідчать, що подібний процес можна використовувати для різних стандартів, а отже, висновки роботи можуть бути використані для подальших досліджень.

Стандарти, використані в каркасі, також досить добре узгоджуються із загальними галузевими рекомендаціями, встановленими Європейською банківською владою. Контрольні пункти CIS пропонують багато заходів безпеки, які безпосередньо пов’язані з більшістю вимог інформаційної безпеки, встановлених EBA. Таким чином, дотримання контрольних заходів CIS допомагає врахувати ширші галузеві вимоги безпеки. Однак для більш повної відповідності рекомендується використовувати інші галузеві стандарти, такі як серія ISO 27000, які надають рекомендації не лише щодо інформаційної безпеки. Проте це виходить за межі роботи та може бути досліджено в майбутньому.

Список використаних джерел

1. Кузьменко О. В., Яровенко Г. М. Сучасні інструменти боротьби з кібершахрайствами у банках. Суми : "Ярославна", 2018. URL: https://essuir.sumdu.edu.ua/bitstream-download/123456789/79743/1/Kuzmenko_%20Kibershakhraistvo_%20paper.pdf
2. Alomari Y., Sulaiman N., Ali S. Explainable Machine Learning for Real-Time Payment Fraud Detection: Building Trustworthy Models to Protect Financial Transactions. *Lecture Notes in Networks and Systems*. 2024. Vol. 1033.
3. Conceptual model of information protection of critical information infrastructure objects of Ukraine / Y. Kozhedub et al. *Collection "Information Technology and Security"*. 2021. Vol. 9, no. 2. P. 151–164. URL: https://doi.org/10.20535/2411-1031.2021.9.2.249889 (дата звернення: 08.11.2024).
4. Evolving strategies in transaction monitoring for enhanced financial security. *FinTech Global*. URL: https://fintech.global/2024/07/24/evolving-strategies-in-transaction-monitoring-for-enhanced-financial-security/.
5. From Risk To Resilience: Harnessing Real-Time Transaction Monitoring. Financial Crime Academy. URL: https://financialcrimeacademy.org/real-time-transaction-monitoring/ (дата звернення: 10.10.2024).
6. Ibitola J. Transforming Transaction Security: Real-Time Monitoring. AI-native AML Compliance & Risk Management Solutions | Flagright. URL: https://www.flagright.com/post/how-real-time-monitoring-changes-the-game-for-transaction-security (дата звернення: 09.10.2024).
7. Prevent Fraud with Real-Time Transaction Monitoring in 2024. Best AML Compliance & Anti-Fraud Software - FOCAL. URL: https://www.getfocal.ai/blog/real-time-transaction-monitoring (дата звернення: 15.10.2024).
8. Real-Time Monitoring Systems For Financial Institutions. BCT Digital. URL: https://www.bctdigital.ai/thought-leadership/real-time-monitoring-systems-in-financial-institutions-challenges-and-solutions/ (дата звернення: 09.10.2024).
9. Redhead M. The Future of Transaction Monitoring: Better Ways to Detect and Disrupt Financial Crime. SSRN Electronic Journal. 2021. URL: https://doi.org/10.2139/ssrn.3865131 (дата звернення: 08.11.2024).
10. Al-Safwani N., Fazea Y., Ibrahim H. ISCP: In-depth model for selecting critical security controls. Computers & Security. 2018. T. 77. C. 565–577. URL: https://doi.org/10.1016/j.cose.2018.05.009 (дата звернення: 02.11.2024).
11. Alshamrani A., Bahattab A. A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. International Journal of Computer Science Issues (IJCSI). 2015. T. 12(1). C. 106–111.
12. Ardo A. A., Bass J. M., Gaber T. Towards Secure Agile Software Development Process: A Practice-Based Model. 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), м. Gran Canaria, Spain, 31 серп. – 2 верес. 2022 р. 2022.

URL: https://doi.org/10.1109/seaa56994.2022.00031 (дата звернення: 02.11.2024).
13. Bonnie E. Understanding Security Frameworks: 14 Common Frameworks Explained. Secureframe. URL: https://secureframe.com/blog/security-frameworks (дата звернення: 02.11.2024).
14. CIS Critical Security Controls FAQ. CIS. URL: https://www.cisecurity.org/controls/cis-controls-faq (дата звернення: 02.11.2024).
15. CIS – Secure Your Business. CIS. URL: https://www.cis-cert.com/en/ (дата звернення: 02.11.2024).
16. Dutta A., Al-Shaer E. “What”, “Where”, and “Why” Cybersecurity Controls to Enforce for Optimal Risk Mitigation. 2019 IEEE Conference on Communications and Network Security (CNS), м. Washington DC, DC, USA, 10–12 черв. 2019 р. 2019. URL: https://doi.org/10.1109/cns.2019.8802745 (дата звернення: 02.11.2024).
17. Evitec Solutions – Inspiring innovations in finance. Evitec Solutions. URL: https://evitec.com/ (дата звернення: 02.11.2024).
18. Force J. Security and Privacy Controls for Information Systems and Organizations : навч. посіб. NIST Special Publication (SP), 2020. 492 с. URL: https://doi.org/10.6028/NIST.SP.800-53r5 (дата звернення: 02.11.2024).
19. framework. Cambridge Dictionary | English Dictionary, Translations & Thesaurus. URL: https://dictionary.cambridge.org/us/dictionary/english/framework (дата звернення: 02.11.2024).
20. Freeman C. Secure Software Development Life Cycle Explained | Black Duck Blog. Synopsys | EDA Tools, Semiconductor IP & Systems Verification. URL: https://www.synopsys.com/blogs/software-security/secure-sdlc.html (дата звернення: 02.11.2024).
21. George T. Semi-Structured Interview | Definition, Guide & Examples. Scribbr. URL: https://www.scribbr.com/methodology/semi-structured-interview/ (дата звернення: 02.11.2024).
22. Guidelines on ICT and security risk management | European Banking Authority. Homepage | European Banking Authority. URL: https://www.eba.europa.eu/guidelines-ict-and-security-risk-management (дата звернення: 02.11.2024).
23. Mapping and Compliance. CIS. URL: https://www.cisecurity.org/cybersecurity-tools/mapping-compliance (дата звернення: 02.11.2024).
24. Team D. E. Thematic Analysis: A Step-by-Step Guide. Customer Insights Hub – Dovetail. URL: https://dovetail.com/research/thematic-analysis/ (дата звернення: 02.11.2024).
25. The most important business risks in 2024: global. Create, find and reuse interactive data visualizations - 23°. URL: https://app.23degrees.io/embed/9k3hfH7Jmf5MfJ8i-bar-horizontal-the-most-important-business (дата звернення: 02.11.2024).

26. Understanding the SDLC: Software Development Lifecycle Explained. GitHub. URL: https://resources.github.com/software-development/what-is-sdlc/ (дата звернення: 02.11.2024).
27. vom Brocke J., Hevner A., Maedche A. Introduction to Design Science Research. Progress in IS. Cham, 2020. С. 1–13. URL: https://doi.org/10.1007/978-3-030-46781-4_1 (дата звернення: 02.11.2024).
28. What Are The Software Development Life Cycle (SDLC) Stages and Models. DistantJob - Remote Recruitment Agency. URL: https://distantjob.com/blog/software-development-life-cycle-stages-and-models/ (дата звернення: 02.11.2024).
29. What Is SDLC? Understand the Software Development Life Cycle - Stackify. Stackify. URL: https://stackify.com/what-is-sdlc/ (дата звернення: 02.11.2024).
30. A Design Science Research Methodology for Information Systems Research / K. Peffers та ін. Journal of Management Information Systems. 2007. Т. 24, № 3. С. 45–77. URL: https://doi.org/10.2753/mis0742-1222240302 (дата звернення: 02.11.2024).
31. Center for Internet Security (CIS) Benchmarks - Microsoft Compliance. Microsoft Learn: Build skills that open doors in your career. URL: https://learn.microsoft.com/en-us/compliance/regulatory/offering-cis-benchmark (дата звернення: 02.11.2024).
32. Consequences of Non-Compliance: Essential Tips To Avoid. Sprinto. URL: https://sprinto.com/blog/consequences-of-non-compliance/ (дата звернення: 02.11.2024).
33. Groš S. A Critical View on CIS Controls. Computer Science. 2019. URL: https://doi.org/10.48550/arXiv.1910.01721 (дата звернення: 02.11.2024).
34. Hennink M., Kaiser B. N. Sample sizes for saturation in qualitative research: A systematic review of empirical tests. Social Science & Medicine. 2022. Т. 292. С. 114523. URL: https://doi.org/10.1016/j.socscimed.2021.114523 (дата звернення: 02.11.2024).
35. Hevner A. A Three Cycle View of Design Science Research. Scandinavian Journal of Information Systems. 2007. Т. 19.
36. How to use the OWASP Top 10 as a standard - OWASP Top 10:2021. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/ (дата звернення: 02.11.2024).
37. Introduction - OWASP Cheat Sheet Series. Introduction - OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/index.html (дата звернення: 02.11.2024).
38. Introduction - OWASP Cheat Sheet Series. Introduction - OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/index.html (дата звернення: 02.11.2024).

39. Khari M., Kumar P., Vaishali. Embedding security in Software Development Life Cycle (SDLC). 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), м. New Delhi. 2016. С. 2182–2186.
40. Kirvan P. Top 12 IT security frameworks and standards explained | TechTarget. Search Security. URL: https://www.techtarget.com/searchsecurity/tip/IT-security-frameworks-and-standards-Choosing-the-right-one (дата звернення: 02.11.2024).
41. Lekh R., Pooja. Exhaustive study of SDLC phases and their best praxctices to create CDP model for process improvement. 2015 International Conference on Advances in Computer Engineering and Applications (ICACEA), м. Ghaziabad, India, 19–20 берез. 2015 р. 2015. URL: https://doi.org/10.1109/icacea.2015.7164852 (дата звернення: 02.11.2024).
42. Manaher S. Model vs Framework: When And How Can You Use Each One?. URL: https://thecontentauthority.com/blog/model-vs-framework (дата звернення: 02.11.2024).
43. Mohan V., Othmane L. B. SecDevOps: Is It a Marketing Buzzword? - Mapping Research on Security in DevOps. 2016 11th International Conference on Availability, Reliability and Security (ARES), м. Salzburg, Austria, 31 серп. – 2 верес. 2016 р. 2016. URL: https://doi.org/10.1109/ares.2016.92 (дата звернення: 02.11.2024).
44. Office for Health Improvement and Disparities. Interview study: qualitative studies. GOV.UK. URL: https://www.gov.uk/guidance/interview-study-qualitative-studies (дата звернення: 02.11.2024).
45. OWASP Application Security Verification Standard 4.0.3. 2021.
46. OWASP Application Security Verification Standard (ASVS) | OWASP Foundation. OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/www-project-application-security-verification-standard/ (дата звернення: 02.11.2024).
47. Ramirez A., Aiello A., Lincke S. J. A Survey and Comparison of Secure Software Development Standards. 2020 13th CMI Conference on Cybersecurity and Privacy (CMI) - Digital Transformation - Potentials and Challenges, м. Copenhagen, Denmark, 26–27 листоп. 2020 р. 2020. URL: https://doi.org/10.1109/cmi51275.2020.9322704 (дата звернення: 02.11.2024).
48. Systematic methodological review: developing a framework for a qualitative semi-structured interview guide / H. Kallio та ін. Journal of Advanced Nursing. 2016. Т. 72, № 12. С. 2954–2965. URL: https://doi.org/10.1111/jan.13031 (дата звернення: 02.11.2024).
49. Team D. E. Semi-Structured Interview: Explanation, Examples, & How-To. Customer Insights Hub – Dovetail. URL: https://dovetail.com/research/semi-structured-interview/ (дата звернення: 02.11.2024).
50. Continuous Security Testing: A Case Study on Integrating Dynamic Security Testing Tools in CI/CD Pipelines / T. Rangnau та ін. 2020 IEEE 24th

International Enterprise Distributed Object Computing Conference (EDOC), м. Eindhoven, Netherlands, 5–8 жовт. 2020 р. 2020. URL: https://doi.org/10.1109/edoc49727.2020.00026 (дата звернення: 02.11.2024).
51. How to deploy a comprehensive DevSecOps solution. Red Hat - We make open source technologies for the enterprise. URL: https://www.redhat.com/en/resources/deploy-comprehensive-devsecops-solution-overview (дата звернення: 02.11.2024).
52. Ramirez A., Aiello A., Lincke S. J. A Survey and Comparison of Secure Software Development Standards. 2020 13th CMI Conference on Cybersecurity and Privacy (CMI) - Digital Transformation - Potentials and Challenges, м. Copenhagen, Denmark, 26–27 листоп. 2020 р. 2020. URL: https://doi.org/10.1109/cmi51275.2020.9322704 (дата звернення: 02.11.2024).
53. Raturi A., Kumar S., Joshi A. Security Risk Assessment & Mitigation Framework for Cloud-based IT Systems. 2022 3rd International Conference on Computing, Analytics and Networks (ICAN), м. Rajpura, Punjab, India, 18–19 листоп. 2022 р. 2022. URL: https://doi.org/10.1109/ican56228.2022.10007263 (дата звернення: 02.11.2024).
54. Schreiber A., Sonnekalb T., Kurnatowski L. v. Towards Visual Analytics Dashboards for Provenance-driven Static Application Security Testing. 2021 IEEE Symposium on Visualization for Cyber Security (VizSec), м. New Orleans, LA, USA, 27 жовт. 2021 р. 2021. URL: https://doi.org/10.1109/vizsec53666.2021.00010 (дата звернення: 02.11.2024).
55. SDLC - Agile Model. URL: https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm (дата звернення: 02.11.2024).
56. Sgobba T. Policy standards and technical standards development and use. Journal of Space Safety Engineering. 2022. URL: https://doi.org/10.1016/j.jsse.2022.07.005 (дата звернення: 02.11.2024).
57. Sönmez F. Ö. Security Qualitative Metrics for Open Web Application Security Project Compliance. Procedia Computer Science. 2019. Т. 151. С. 998–1003. URL: https://doi.org/10.1016/j.procs.2019.04.140 (дата звернення: 02.11.2024).
58. S S. A Study of Software Development Life Cycle Process Models. SSRN Electronic Journal. 2017. URL: https://doi.org/10.2139/ssrn.2988291 (дата звернення: 02.11.2024).
59. Tan V., Cheh C., Chen B. From Application Security Verification Standard (ASVS) to Regulation Compliance: A Case Study in Financial Services Sector. 2021 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), м. Wuhan, China, 25–28 жовт. 2021 р. 2021. URL: https://doi.org/10.1109/issrew53611.2021.00046 (дата звернення: 02.11.2024).

60. Tissir N., El Kafhali S., Aboutabit N. Cybersecurity management in cloud computing: semantic literature review and conceptual framework proposal. Journal of Reliable Intelligent Environments. 2020. URL: https://doi.org/10.1007/s40860-020-00115-0 (дата звернення: 02.11.2024).
61. Venable J., Pries-Heje J., Baskerville R. FEDS: a Framework for Evaluation in Design Science Research. European Journal of Information Systems. 2016. Т. 25, № 1. С. 77–89. URL: https://doi.org/10.1057/ejis.2014.36 (дата звернення: 02.11.2024).
62. Wen S.-F., Katt B. A quantitative security evaluation and analysis model for web applications based on OWASP application security verification standard. Computers & Security. 2023. Т. 135. С. 103532. URL: https://doi.org/10.1016/j.cose.2023.103532 (дата звернення: 02.11.2024).
63. Top 11 cybersecurity frameworks in 2024. MSP Technology and IT Management Software | ConnectWise. URL: https://www.connectwise.com/blog/cybersecurity/11-best-cybersecurity-frameworks (дата звернення: 02.11.2024).
64. Roy P. P. A High-Level Comparison between the NIST Cyber Security Framework and the ISO 27001 Information Security Standard. 2020 National Conference on Emerging Trends on Sustainable Technology and Engineering Applications (NCETSTEA), м. Durgapur, India, 7–8 лют. 2020 р. 2020. URL: https://doi.org/10.1109/ncetstea48365.2020.9119914 (дата звернення: 02.11.2024).

Додаток 1. Керівництво ЕВА щодо управління ризиками ІСТ та безпеки [22]

Керівництво щодо управління ризиками ІСТ та безпеки					
Розділ	Тип	Категорія	Опис	Засоби CIS	Вимоги OWASP ASVS
3.2.1	Управління	Управління	Забезпечити наявність належного внутрішнього управління та внутрішнього контрольного каркасу для ІСТ і ризиків безпеки.	-	--
3.2.1	Управління	Навчання	Забезпечити, щоб усі співробітники, включаючи тих, хто відповідає за критичні функції, проходили належне навчання щодо ІСТ і ризиків безпеки, включаючи інформаційну безпеку, щорічно або частіше, якщо це необхідно.	14.X, 16.9	-
3.3.1	Організація та цілі	Управління	Функції ІСТ, відповідальні за системи ІСТ, процеси та операції безпеки, повинні мати відповідні процеси та контролі для забезпечення, що всі ризики ідентифіковані, проаналізовані, виміряні, моніторяться, керуються, звітуються та	7.X, 17.X	-

			утримуються в межах встановленого ризикового апетиту фінансової установи. А також, що проєкти, системи та діяльність, які вони виконують, відповідають зовнішнім і внутрішнім вимогам.		
Інформаційна безпека					
3.4.1	Політика інформаційної безпеки	Управління	Розробка та документування політики інформаційної безпеки, яка визначає основні принципи та правила для захисту конфіденційності, цілісності та доступності даних і інформації фінансових установ та їх клієнтів.	-	-
3.4.1	Політика інформаційної безпеки	Відповідальність	Визначення основних ролей і обов'язків управління інформаційною безпекою.	-	-
3.4.1	Політика інформаційної безпеки	Відповідальність	Визначення вимог до персоналу, підрядників, процесів і технологій.	-	-

3.4.1	Політика інформаційної безпеки	Навчання	Визнання, що весь персонал і підрядники несуть відповідальність за забезпечення інформаційної безпеки.	14.X	-
3.4.1	Політика інформаційної безпеки	Дані	Забезпечення конфіденційності, цілісності та доступності критичних активів, ресурсів і конфіденційних даних.	-	v1.6, v1.8, v4.X.X, v8.X.X, v9.X.X, v13.1.2, v13.1.3, v13.2.6
3.4.2	Логічна безпека	Права доступу	Управління правами доступу на основі принципу «необхідності знати».	-	-
3.4.2	Логічна безпека	Права доступу	Надання користувачам мінімально необхідних прав доступу для виконання їхніх обов’язків (принцип «мінімальних привілеїв»).	6.1, 6.2	v4.1.3, v4.1.4
3.4.2	Логічна безпека	Права доступу	Забезпечення розподілу обов'язків для запобігання несанкціонованому доступу або неправомірного призначення комбінацій прав доступу.	6.8	v1.2.2, v1.4.4, v1.4.5, v1.4.14, v4.X.X
3.4.2	Логічна безпека	Відповідальність	Обмеження використання загальних і	5.1	v2.5.4

			спільних облікових записів користувачів.		
3.4.2	Логічна безпека	Логування	Забезпечення можливості ідентифікації дій користувачів в ІСТ-системах.	8.X	v7.2.X
3.4.2	Логічна безпека	Права доступу	Реалізація суворих заходів контролю над привілейованим доступом до системи.	5.4	v1.2.1, v1.2.2, v1.4.3
3.4.2	Логічна безпека	Права доступу	Суворе обмеження та контроль облікових записів з підвищеними правами доступу до системи.	5.1	v1.2.1
3.4.2	Логічна безпека	Права доступу	Надання віддаленого адміністративного доступу до критичних ІСТ-систем тільки на основі «необхідності знати» із використанням сильних рішень автентифікації.	6.1, 6.4, 6.6	-
3.4.2	Логічна безпека	Логування	Логування та моніторинг усіх дій привілейованих користувачів.	3.14, 8.X	v1.7, v7.X.X
3.4.2	Логічна безпека	Логування	Захист логів доступу для запобігання несанкціонованій модифікації або видаленню.	8.3	v1.7, v7.X.X

3.4.2	Логічна безпека	Логування	Збереження логів протягом періоду, що відповідає критичності бізнес-функцій.	8.X	v1.7
3.4.2	Логічна безпека	Права доступу	Надання, скасування або модифікація прав доступу без зволікань.	6.1, 6.2, 6.7	-
3.4.2	Логічна безпека	Права доступу	Виконання заздалегідь визначених процесів затвердження, що включають власника інформаційного активу.	-	-
3.4.2	Логічна безпека	Права доступу	Своєчасне скасування прав доступу після припинення трудових відносин.	6.1, 6.2, 6.7	-
3.4.2	Логічна безпека	Права доступу	Періодичний перегляд прав доступу для забезпечення відсутності надмірних привілеїв.	6.1, 6.2, 6.7	-
3.4.2	Логічна безпека	Права доступу	Скасування прав доступу, коли вони більше не потрібні.	6.2	-
3.4.2	Логічна безпека	Автентифікація	Застосування методів автентифікації відповідно до рівня доступу з критичністю систем ІСТ.	5.X, 6.X	v1.2, v2.X.X, v4.X.X
3.4.2	Логічна безпека	Автентифікація	Включення складних паролів	5.1, 5.2	v2.X.X

			або більш сильних методів автентифікації відповідно до відповідного ризику.		
3.4.2	Логічна безпека	Права доступу	Обмеження електронного доступу додатками до мінімального, необхідного для надання відповідних послуг.	-	v4.X.X
3.4.4	Безпека роботи ІСТ	Запобігання	Впровадження процедур для запобігання проблемам із безпекою в ІСТ-системах та послугах.	-	-
3.4.4	Безпека роботи ІСТ	Запобігання	Оцінка та усунення потенційних вразливостей.	7.X, 16.2, 16.6, 17.X	1.10.1
3.4.4	Безпека роботи ІСТ	Оновлення	Забезпечення актуальності програмного забезпечення та мікропрограм.	12.1, 13.5, 16.5	v1.14.3
3.4.4	Безпека роботи ІСТ	Оновлення	Розгортання критичних виправлень безпеки або впровадження компенсуючих заходів контролю.	12.1, 13.5, 16.5	-
3.4.4	Безпека роботи ІСТ	Конфігурація	Впровадження безпечних базових конфігурацій для всіх мережевих компонентів.	4.2	v1.14.X, v14.X.X

3.4.4	Безпека роботи ІСТ	Конфігурація	Впровадження сегментації мережі.	12.2, 13.4	v1.14.5
3.4.4	Безпека роботи ІСТ	Дані	Впровадження системи запобігання втраті даних.	3.13	-
3.4.4	Безпека роботи ІСТ	Шифрування	Шифрування мережевого трафіку відповідно до класифікації даних.	3.10, 12.6	v13.2.6, v13.3.X
3.4.4	Безпека роботи ІСТ	Захист	Захист кінцевих точок, включаючи сервери, робочі станції та мобільні пристрої.	-	-
3.4.4	Безпека роботи ІСТ	Права доступу	Оцінка стандартів безпеки кінцевих точок перед наданням доступу до корпоративної мережі.	-	-
3.4.4	Безпека роботи ІСТ	Права доступу	Забезпечення механізмів перевірки цілісності програмного забезпечення, мікропрограм та даних.	-	-
3.4.4	Безпека роботи ІСТ	Шифрування	Шифрування даних у спокої та під час передачі відповідно до класифікації даних.	3.10, 3.11	v6.1.X, v13.2.X
3.4.5	Моніторинг безпеки	Моніторинг	Постійне виявлення та моніторинг.	7.5, 7.6, 10.X, 13.X	v1.2.3, v10.X.X, v11.1.4, v11.1.8, v14.1.5

3.4.5	Моніторинг безпеки	Моніторинг	Впровадження заходів для виявлення витоків інформації, шкідливого коду та інших загроз безпеці.	4.X, 7.5, 7.6, 10.X, 13.X, 16.12	v1.10.X, v10.X
3.4.5	Моніторинг безпеки	Моніторинг	Моніторинг публічно відомих вразливостей у програмному та апаратному забезпеченні та застосування відповідних оновлень	16.4, 16.5	-
3.4.6	Оцінка та тестування інформаційної безпеки	Запобігання	Розгляд перевірок вихідного коду.	16.1	-
3.4.6	Оцінка та тестування інформаційної безпеки	Запобігання	Розгляд оцінки вразливостей.	7.X, 16.1, 16.2, 16.3, 16.13, 16.6, 16.14	-
3.4.6	Оцінка та тестування інформаційної безпеки	Тестування	Розгляд тестування на проникнення.	16.13, 18.X	-
3.4.6	Оцінка та тестування інформаційної безпеки	Навчання	Розгляд навчань команди червоного (Red Team).	14.X, 16.9	-
3.4.6	Оцінка та тестування інформаційної безпеки	Тестування	Встановлення основ тестування інформаційної безпеки.	16.13, 18.X	-
3.4.6	Оцінка та тестування інформаційної безпеки	Моніторинг	Моніторинг та оцінка результатів тестів безпеки з подальшим оновленням	-	-

			заходів безпеки у критичних системах ІСТ.		
3.4.7	Навчання та обізнаність з інформаційної безпеки	Навчання	Запровадження програми навчання.	14.X, 16.9	-
Менеджмент операцій в ІСТ					
3.5	Управління операціями ІСТ	Управління	Управління операціями ІСТ на основі задокументованих та впроваджених процесів і процедур.	-	-
3.5	Управління операціями ІСТ	Управління	Підтримка актуального інвентарю ресурсів ІСТ.	1.X	-
3.5	Управління операціями ІСТ	Моніторинг	Реалізація процедур логування та моніторингу для критичних операцій ІСТ з метою виявлення, аналізу та виправлення помилок.	3.14, 8.X, 9.X, 10.X, 13.X	v1.7, v7.X.X
3.5	Управління операціями ІСТ	Дані	Визначення та реалізація процедур резервного копіювання та відновлення даних і систем ІСТ для забезпечення їх відновлення за потреби.	11.1, 11.2, 11.3, 11.4	v8.1.5, v8.1.6, v14.1.4
3.5	Управління операціями ІСТ	Дані	Забезпечити безпечне зберігання резервних копій даних та систем	11.1, 11.3, 11.4	v8.1.5, v8.1.6

			ІСТ, достатньо віддалених від основного сайту, щоб уникнути однакових ризиків.		
3.5	Управління операціями ІСТ	Моніторинг	Моніторинг підтримки зовнішніх або внутрішніх постачальників для їхніх ресурсів ІСТ.	15.X	-
3.5	Управління операціями ІСТ	Моніторинг	Забезпечити застосування всіх відповідних оновлень та апгрейдів на основі задокументованих процесів.	2.1, 2.2	v10.3.X
3.5.1	Управління інцидентами ІСТ	Управління	Реалізація процесів та організаційних структур для моніторингу, обробки та контролю інцидентів безпеки та операційних інцидентів.	17.X	-
Управління проектами та змінами в ІСТ					
3.6.2	Придбання та розробка систем	Тестування	Розробити методологію тестування та затвердження систем ІСТ перед їх першим використанням.	16.13, 18.X	v1.1.1
3.6.2	Придбання та розробка систем	Тестування	Тестувати системи ІСТ, сервіси ІСТ та заходи інформаційної безпеки для виявлення	11.5, 16.13, 18.X	v1.1.2

			потенційних слабкостей, порушень і інцидентів.		
3.6.2	Придбання та розробка систем	Конфігурація	Реалізувати окремі середовища ІСТ для забезпечення адекватного розподілу обов'язків і зменшення впливу неперевірених змін у продуктивних системах.	16.8	v1.14.1, v4.3.3
3.6.2	Придбання та розробка систем	Дані	Забезпечити цілісність і конфіденційність виробничих даних у непроизводних середовищах. Доступ до виробничих даних обмежений авторизованими користувачами.	3.1, 3.7, 3.12, 3.14, 6.6, 6.8	-
3.6.2	Придбання та розробка систем	Захист	Реалізувати заходи для захисту цілісності вихідного коду систем ІСТ, що розробляються внутрішньо.	1.1, 2.X, 3.1, 3.3, 4.X, 5.8, 6.X, 7.X, 13.X, 16.12	1.10.1, 10.2.X
Управління безперервністю бізнесу					
3.7.3	План реагування та відновлення	Дані	Розробити плани реагування та відновлення.	11.X	-

Архітектура сервісу онлайн-моніторингу для захисту банківських активів клієнта

Виконав: Горбар Олександр Едуардович

Керівник: Сагайдак Віктор Анатолійович

АКТУАЛЬНІСТЬ ТЕМИ

- Ландшафт кібербезпеки постійно змінюється, що створює нові виклики для організацій.
- Сучасні регуляції, такі як GDPR, вимагають високого рівня відповідності стандартам безпеки.
- Інтеграція численних стандартів у процес розробки програмного забезпечення залишається складним завданням.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета:** Об'єднання рекомендацій з безпеки застосунків та інфраструктури в єдиний програмний каркас.
- **Завдання:**
  - Вивчення стандартів OWASP ASVS та CIS Controls.
  - Розробка уніфікованого каркасу.
  - Оцінка відповідності каркасу потребам фінансової індустрії.
- **Об'єкт:** Процес побудови програмного каркасу безпеки.
- **Предмет:** Архітектура безпеки програмного забезпечення.

ОСНОВНІ ДОСЛІДНИЦЬКІ ПИТАННЯ

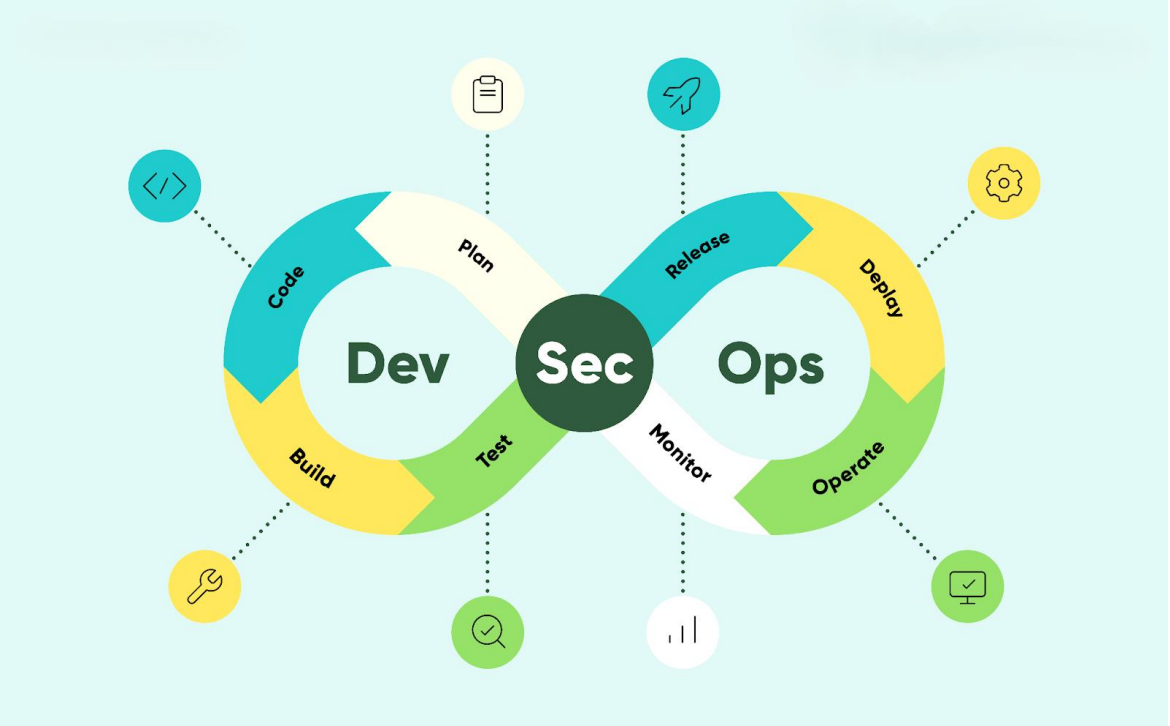
- Як розробити програмний каркас, що інтегрує безпеку в SDLC?
- Наскільки каркас відповідає специфічним вимогам фінансової індустрії?

Питання	Очікуваний результат
Як інтегрувати безпеку в SDLC?	Розроблений каркас із рекомендаціями
Чи відповідає каркас вимогам?	Оцінка експертами з фінансової галузі

Таблиця: Ключові запитання до галузевих експертів та очікувані результати.

ПРОГРАМНИЙ КАРКАС БЕЗПЕКИ

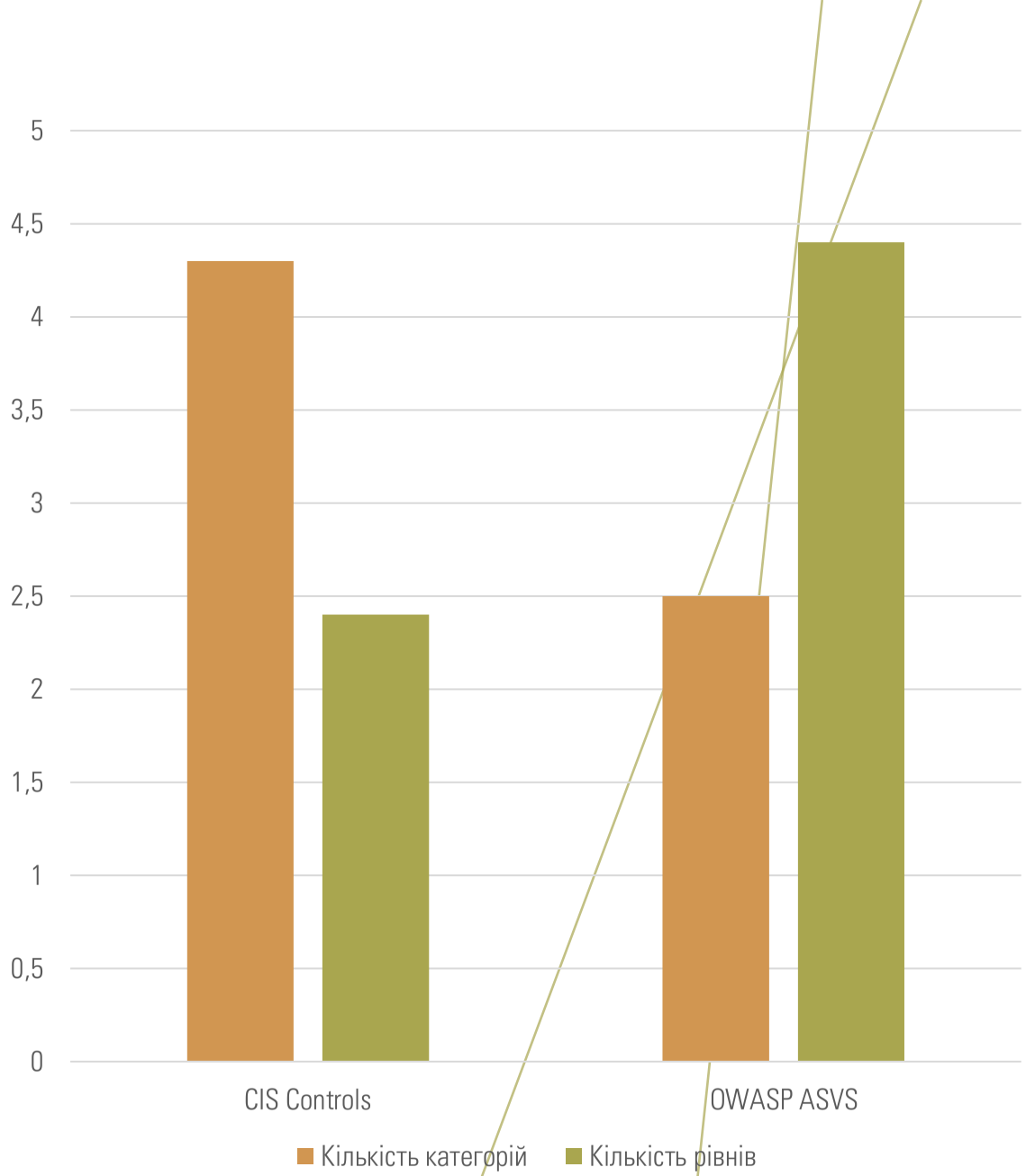
- Розроблений каркас інтегрує рекомендації OWASP ASVS та CIS Controls.
- Він виступає як контрольний список, який допомагає організаціям дотримуватись регуляторних вимог.



Візуалізація: Схема структури каркасу.

АНАЛІЗ СТАНДАРТІВ БЕЗПЕКИ

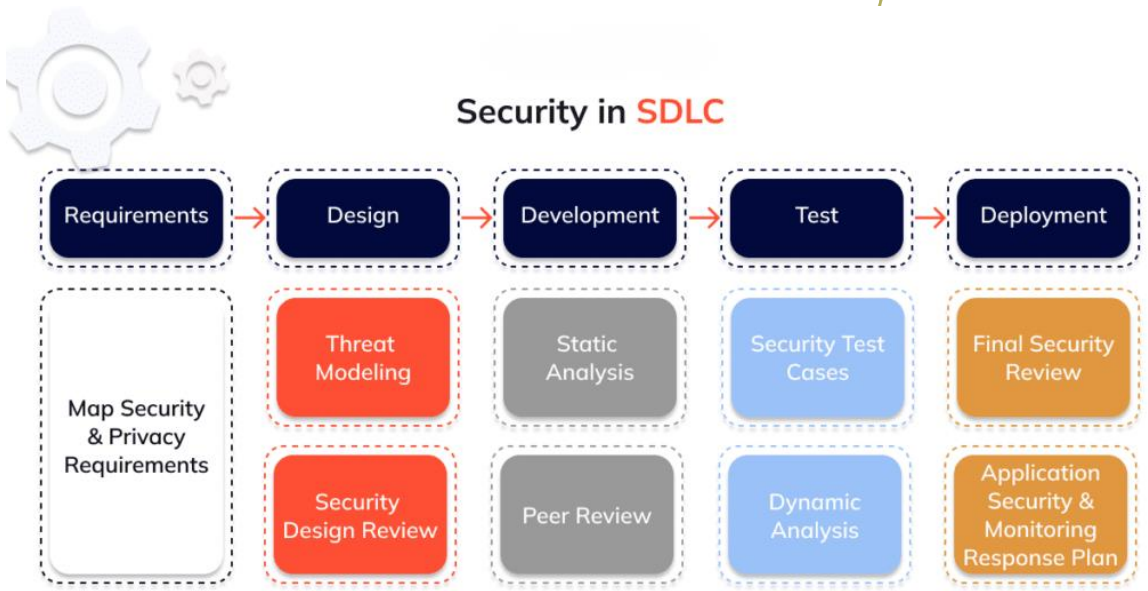
- CIS Controls:
  - 18 заходів, поділених на три групи впровадження.
  - Спрямовані на управління ризиками та захист даних.
- OWASP ASVS:
  - 14 категорій вимог і три рівні перевірки безпеки.



Графік: Порівняння рівнів впровадження.

ЖИТТЄВИЙ ЦИКЛ РОЗРОБКИ *(SDLC)*

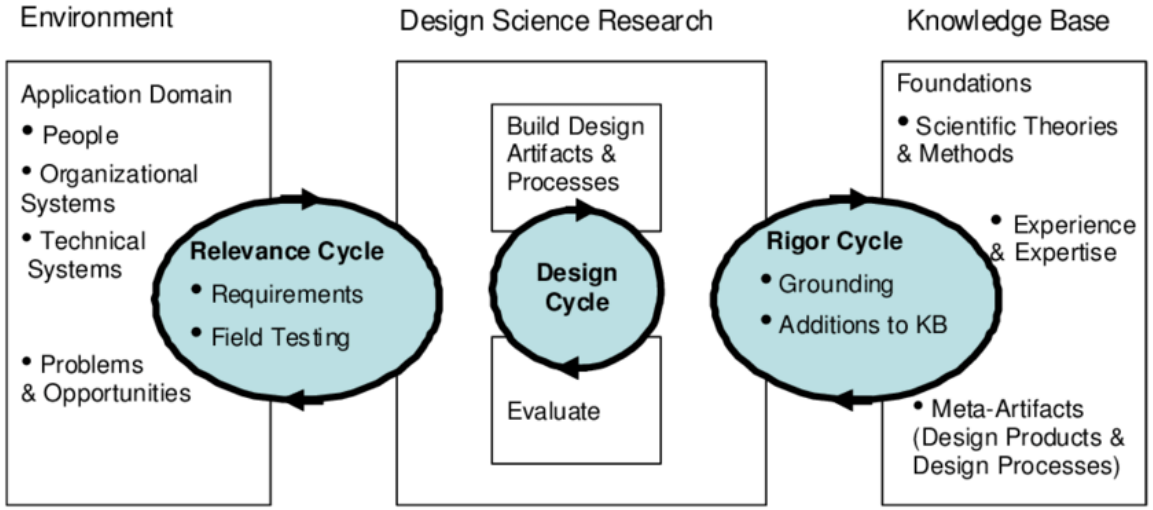
- Життєвий цикл включає етапи від планування до підтримки.
- Інтеграція заходів безпеки забезпечує відповідність стандартам на кожному етапі.



Додаток: Блок-схема SDLC із зазначенням заходів безпеки.

МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

- Для розробки каркасу використано підхід Design Science Research.
- Основні етапи:
  - Ідентифікація проблеми.
  - Розробка артефакту.
  - Оцінка через експертні інтерв'ю.



Візуалізація: Цикли Design Science Research.

ОЦІНКА ПРОГРАМНОГО КАРКАСУ

- Експертні інтерв'ю показали, що каркас спрощує інтеграцію стандартів безпеки в SDLC.
- Каркас відповідає вимогам фінансової індустрії, забезпечуючи якість захисту.

ПРАКТИЧНА ЗНАЧУЩІСТЬ

- Каркас спрощує управління безпекою та підвищує відповідність регуляціям.
- Його застосування підвищує якість безпеки програмного забезпечення.

НАУКОВА НОВИЗНА

- Додано фазу "Загальні вимоги" до SDLC, яка враховує організаційні аспекти безпеки. Потенційний вплив:
  - Зниження кількості вразливостей на ранніх етапах розробки.
  - Зменшення часу та витрат на тестування безпеки на пізніх етапах.
  - Підвищення узгодженості вимог безпеки між командами розробників та аналітиків.
  - Забезпечення відповідності регуляторним вимогам ще на етапі проєктування.
- Каркас демонструє ефективність мапінгу безпекових вимог.

ВИСНОВКИ

- Досягнуті результати:
  - Розроблено уніфікований програмний каркас, що об'єднує рекомендації OWASP ASVS і CIS Controls.
  - Проведено аналіз стандартів безпеки та їхню інтеграцію в життєвий цикл розробки.
  - Каркас підтвердив свою ефективність у рамках фінансової індустрії.
- Основні переваги:
  - Забезпечує відповідність регуляторним вимогам.
  - Інтегрує заходи безпеки на всіх етапах SDLC.
  - Полегшує роботу розробників і підвищує загальний рівень безпеки.
- Перспективи:
  - Тестування каркасу в реальних проєктах різних галузей.
  - Розширення функціональності для покриття додаткових вимог стандартів.
  - Вдосконалення методик оцінки ефективності каркасу.

АПРОБАЦІЯ

II ВСЕУКРАЇНСЬКА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ ТЕХНОЛОГІЧНІ ГОРИЗОНТИ: ДОСЛІДЖЕННЯ ТА ЗАСТОСУВАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ ТЕХНОЛОГІЧНОГО ПРОГРЕСУ УКРАЇНИ І СВІТУ

- Представлено проблематику та обґрунтування необхідності розробки рішення
- Запропоновано багатошарову архітектуру: збір даних, обробку даних, аналітику, інтерфейси для операторів та захист від зовнішніх атак

ПОДЯКА

Дякую за увагу!

Вдячний керівнику за підтримку!