Міністерство освіти і науки України Одеський національний технологічний університет
Кафедра комп'ютерної інженерії

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему *Дослідження процесів розробки та впровадження програмного забезпечення за допомогою Devops методології*

(назва кваліфікаційної роботи згідно наказу ОНТУ)

Здобувача *Зінченка С.В*
(прізвище, ініціали)

2 курсу *777* групи

Керівники: *ст.викл. Сіренко О.І*
(посада, прізвище та ініціали)

д.е.н., проф., Артеменко С.В
(посада, прізвище та ініціали)

Консультанти:
(посада, прізвище та ініціали)

д.е.н., проф. Басюркіна Н.Й.
(посада, прізвище та ініціали)

Кваліфікаційна робота допускається до захисту

Рішення кафедри від *30.11* 2023 р., протокол № 3

Завідувач кафедри комп. інженерії _____ Сергій АРТЕМЕНКО
(назва кафедри) (підпис) (Ім'я ПРІЗВИЩЕ)

Одеса – 2023 рік

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерної інженерії, програмування та кіберзахисту* Кафедра *комп'ютерної інженерії*
Ступінь вищої освіти *магістр*
Спеціальність *123 «Комп'ютерна інженерія»*
Освітня програма *Спеціалізовані комп'ютерні системи*

ЗАТВЕРДЖУЮ
Зав. кафедри *комп'ютерної інженерії*
Сергій АРТЕМЕНКО
«*30*» листопада 2022 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Зінченка Сергія Вікторовича

1. Тема роботи *Дослідження процесів розробки та впровадження програмного забезпечення за допомогою Devops методології*

Затверджена наказом університету від « *30* » *листопада* 2022 р., наказ № *884-03*

2 Термін здачі здобувачем закінченої роботи *28 листопада 2023 р.*

3. Вихідні дані роботи

Огляд основних підходів, методів та існуючих інструментів неперервної інтеграції коду та його розгортання, порівняльний аналіз провідних інструментів, побудова проекту на основі проведеного аналізу. Технічна документація DevSecOps.

4. Перелік питань, які потрібно розробити

1. Вступ. 2. Аналіз методологій розробки програмного забезпечення. 3.Розробка методики ризик-орієнтованого тестування. 4.Результати тестування та оцінка ризиків. 5. Екогомічна частина. 6. Охорона праці.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Економіка	*д.е.н., проф. Басюркіна Н.Й.*		
Охорона праці	*д.т.н., проф., Артеменко С.В*		
Нормоконтроль	*ст. викл., Сіренко О.І*		

7. Дата видачі завдання *30.11.2022*

Керівники *Сергій АРТЕМЕНКО*

Олександр СІРЕНКО

Завдання прийняв до виконання *Сергій ЗІНЧЕНКО*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1.	*Дослідженні предметної області*	*01.07.2023*	
2.	*Дослідження складових елементів Devops*	*28.07.2023*	
3.	*Дослідження методів керування проектами*	*25.08.2023*	
4.	*Розробка методики тестування*	*15.09.2023*	
5.	*Проведення тестування*	*1.10.2023*	
6.	*Техніко-економічний аналіз проекту*	*30.10.2023*	
7.	*Розробка розділу з охорони праці*	*10.11.2023*	
8.	*Оформлення пояснювальної записки*	*27.11.2023*	

Здобувач-дипломник *Сергій ЗІНЧЕНКО*

Керівники роботи *Сергій АРТЕМЕНКО*

Олександр СІРЕНКО

Несу відповідальність за ідентичність електронного та друкованого варіантів кваліфікаційної роботи, даю згоду на обробку персональних даних та не заперечую проти розміщення кваліфікаційної роботи на офіційних web-ресурсах ОНТУ.

Підтверджую, що в кваліфікаційній роботі відсутні порушення норм академічної доброчесності.

Здобувач-дипломник *Сергій ЗІНЧЕНКО*

АНОТАЦІЯ

Автоматизоване тестування та постійна доставка допомагають зменшити ризики, пов'язані з впровадженням нових функцій чи змінами в інфраструктурі. Узагальнюючи, DevOps залишається ключовою стратегією для компаній, які прагнуть покращити ефективність своєї розробки та впровадження програмного забезпечення, особливо в умовах швидкозмінюваного технологічного середовища.

Робота має за мету дослідити процеси розробки та впровадження програмного забезпечення за допомогою Devops методології. Здійснено аналіз методологій розробки програмного забезпечення. Окреслено розробку методики ризик-орієнтованого тестування. Досліджено результати тестування та оцінку ризиків. Проведено розрахунки економічної ефективності впровадження нового програмного продукту.

Практичне значення роботи полягає в тому що результати можуть використовуватися для управління ризиками кібербезпеки та тестування методикою ризик-орієнтованого тестування. Наукова новизна дослідження: управління ризиками кібербезпеки з використанням методу аналізу та оцінки ризиків, який має статус оптимального за визначеними критеріями.

Ключові слова: програмне забезпечення, кібербезпека, управління ризиками, технологічне середовище, ризик-орієнтоване тестування

ABSTRACT

Automated testing and Continuous Delivery help reduce the risks associated with new features or infrastructure changes. Overall, DevOps remains a key strategy for companies looking to improve the efficiency of their software development and implementation, especially in a rapidly changing technology environment.

The aim of the paper is to explore the processes of software development and implementation using the Devops methodology. Software development methodologies are analyzed. The development of a risk-based testing methodology is outlined. The results of testing and risk assessment were studied. Calculations of the economic efficiency of implementing a new software product are carried out.

The practical significance of the work lies in the fact that the results can be used for cybersecurity risk management and testing using risk-based testing methods. Scientific novelty of the study: cybersecurity risk management using the risk analysis and assessment method, which has the status of optimal according to certain criteria.

Keywords: software, cybersecurity, risk management, technology environment, risk-based testing

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1 АНАЛІЗ МЕТОДОЛОГІЙ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 10
1.1 Методологія DevOps 10
1.2 Методологія Agile 13
1.3 Методологія Waterfall 14
1.4 Завдання безпеки в DevOps 15
1.5 Архітектура середовища безпечної розробки різновиду DevSecOps 16
Висновки до першого розділу 18
РОЗДІЛ 2 РОЗРОБКА МЕТОДИКИ РИЗИК-ОРІЄНТОВАНОГО ТЕСТУВАННЯ 20
2.1 Вимоги до процесу тестування в DevOps 20
2.2 Ризик-орієнтований підхід до тестування 21
2.3 Вибір методів оцінки ризиків 24
2.4 Методика ризик-орієнтованого тестування в конвеєрі DevSecOps 32
2.5 Моделювання управління ризиками з використанням розробленої методики 35
Висновки до другого розділу 38
РОЗДІЛ 3 РЕЗУЛЬТАТИ ТЕСТУВАННЯ ТА ОЦІНКА РИЗИКІВ 40
3.1 Створення програмного комплексу для оцінки ризиків 40
3.2 Результати проведення ризик-орієнтованого тестування 44
Висновки до третього розділу 51
РОЗДІЛ 4 ЕКОНОМІЧНА ЧАСТИНА 53
4.1 Організаційно-економічне та маркетингове обґрунтування проекту 53

4.2 Розрахунки економічної ефективності впровадження нового програмного продукту 57
4.3 Визначення капітальних витрат 59
Висновки до четвертого розділу 60
РОЗДІЛ 5 ОХОРОНА ПРАЦІ 62
5.1 Аналіз умов праці на робочому місці програміста 62
5.2 Промислова безпека у виробничому приміщенні 66
5.3 Виробнича санітарія у приміщенні 67
5.4 Пожежна безпека виробничого приміщення 69
Висновки до п'ятого розділу 71
ЗАГАЛЬНІ ВИСНОВКИ 72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 74
ДОДАТКИ 79
Додаток А Лістинг коду Jenkins Pipeline для CI/CD бекенд проекту 79
Додаток Б Графічний матеріал 83

ВСТУП

Станом на кінець 2022 року та початок 2023 року методологія DevOps залишається дуже актуальною в області розробки програмного забезпечення. Велика кількість компаній в різних галузях вже успішно впроваджують DevOps для поліпшення ефективності своїх IT-процесів.

DevOps сприяє автоматизації тестування, інтеграції та впровадження, що дозволяє розробникам швидше створювати та впроваджувати новий функціонал. Автоматизовані тести, постійна інтеграція і постійна доставка (CI/CD) дозволяють виявляти та виправляти помилки на ранніх етапах розробки, підвищуючи якість продукту.

Використання інфраструктурного коду (Infrastructure as Code, IaC) дозволяє автоматизувати розгортання та конфігурування інфраструктури, забезпечуючи консистентність середовищ. DevOps допомагає зменшити час виявлення та усунення помилок завдяки автоматизації тестування та швидкому впровадженню змін.

DevOps вирішує проблеми, пов'язані з комунікацією та співпрацею між розробниками та операторами, стимулюючи більш ефективне спілкування та співпрацю. DevOps дозволяє легко масштабувати інфраструктуру та додавати новий функціонал, що є критичним для компаній, які стрімко ростуть.

Автоматизоване тестування та постійна доставка допомагають зменшити ризики, пов'язані з впровадженням нових функцій чи змінами в інфраструктурі. Узагальнюючи, DevOps залишається ключовою стратегією для компаній, які прагнуть покращити ефективність своєї розробки та впровадження програмного забезпечення, особливо в умовах швидкозмінюваного технологічного середовища.

Мета дослідження – дослідити процеси розробки та впровадження програмного забезпечення за допомогою Devops методології.

Відповідно до мети роботи необхідно вирішити наступні *завдання:*

1. Здійснити аналіз методологій розробки програмного забезпечення.

2. Окреслити розробку методики ризик-орієнтованого тестування.

3. Дослідити результати тестування та оцінку ризиків.

4. Провести розрахунки економічної ефективності впровадження нового програмного продукту.

Об'єкт дослідження – інформаційна безпека технології розробки програмного забезпечення DevOps.

Предмет дослідження – процеси розробки та впровадження програмного забезпечення за допомогою Devops методології.

Методами дослідження було обрано: аналіз архітектурних особливостей і ризиків безпеки в DevOps, використання існуючих методи оцінки та аналізу ризиків, методики ризик-орієнтованого тестування.

Практичне значення роботи полягає в тому що результати можуть використовуватися для управління ризиками кібербезпеки та тестування методикою ризик-орієнтованого тестування.

Наукова новизна дослідження: управління ризиками кібербезпеки з використанням методу аналізу та оцінки ризиків, який має статус оптимального за визначеними критеріями.

Структура роботи. Дипломна робота складається зі вступу, чотирьох розділів основної частини, висновків та списку використаних джерел.

РОЗДІЛ 1
АНАЛІЗ МЕТОДОЛОГІЙ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Методологія DevOps

Впровадження програмного забезпечення – це процес виходу програмного продукту на ринок. Впровадженням називають доставку вашого продукту до кінцевого користувача.

DevOps (акронім від англ. development і operations) – низка практик, призначених для пожвавлення взаємодії розробників із фахівцями інформаційно- технологічного обслуговування та зближення (рис.1.1) їхніх робочих процесів одне за одним.

Рис. 1.1 – Основні одиниці DevOps

DevOps базується на тісній залежності між розробкою та використанням ПЗ, що знаходиться на стику 3-х складових (рис. 1.2), допомагає організаціям швидше створювати і оновлювати програмні продукти та послуги.

Рис. 1.2 – Складові DevOps

DevOps – це культура, яка сприяє взаємодії між розробницькою та

операційною командами, спрощуючи та прискорюючи впровадження коду в продуктивне середовище за допомогою автоматизації [1]. Далі розглянемо різні етапи життєвого циклу DevOps, як показано на рис. 1.3.:

– планування. Цей етап допомагає визначити цінність продукції для бізнесу та встановлює вимоги;

– код. На цьому етапі відбувається проектування програмного забезпечення та написання його коду;

– розробка. Керуючи збірками та версіями програмного забезпечення, на цьому етапі використовують автоматизовані інструменти для компіляції пакетного коду для майбутнього випуску. Також використовують репозиторії вихідного коду або репозиторії пакетів, які "упаковують" інфраструктуру, необхідну для випуску продукту;

– тестування. Цей етап включає в себе безперервне тестування, будь то ручне чи автоматичне, спрямоване на досягнення оптимальної якості коду;

– впровадження. Цей етап визначає впровадження коду в операційне середовище з метою надання продукту функціональності та доступності;

– експлуатація. На цьому етапі здійснюється управління програмним забезпеченням під час експлуатації;

– моніторинг. На цьому етапі виконується визначення та збір інформації про проблеми, що здійснюються у конкретній версії програмного забезпечення у процесі експлуатації.

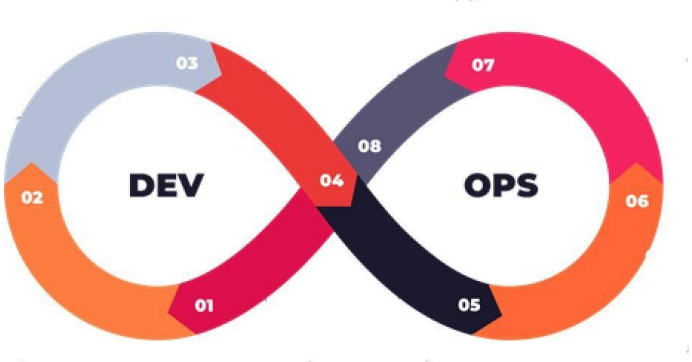


Рис. 1.3 – Етапи методології ДевОпс

Практика DevOps віддзеркалює концепцію безперервного вдосконалення та автоматизації.

DevOps надає ряд переваг, серед яких:

– більш швидке та якісне надання продуктів. Забезпечує швидше впровадження нових функцій та оновлень, а також покращену якість продукту через неперервний контроль якості;

– більш швидке вирішення проблем та зниження складності. Допомагає виявляти та вирішувати проблеми швидше завдяки автоматизованому тестуванню та постійній інтеграції;

– більш висока масштабність і доступність. Забезпечує зручну масштабованість і високу доступність системи завдяки автоматичному масштабуванню та надійній інфраструктурі;

– більш стабільне робоче середовище. Зменшує ризик помилок та конфліктів завдяки автоматизованому управлінню конфігурацією та стандартизації середовищ;

– більш ефективне використання ресурсів. Дозволяє оптимально використовувати інфраструктурні та людські ресурси завдяки автоматизації та управлінню ресурсами;

– більш висока автоматизація. Забезпечує автоматизований процес випуску, тестування та розгортання коду, що прискорює весь життєвий цикл розробки;

– більше наглядового представлення результатів роботи систем. Дозволяє в режимі реального часу моніторити та аналізувати стан і результати роботи системи;

– інновації. Сприяє швидкій впровадженню новаторських ідей та функцій через безперервний процес розробки та впровадження.

1.2 Методологія Agile

Agile - це методологія розробки програмного забезпечення, яка базується на принципах гнучкості, спрямованих на підвищення ефективності та адаптивності розробки продукту. Основні принципи методології Agile визначені в "Маніфесті Agile", який складається з 12 принципів, що підкреслюють важливість співпраці команд, змінливості вимог, та постійного вдосконалення процесів розробки [19].

Основні ідеї та практики методології Agile включають:

1. Ітераційний та інкрементний підхід. Розробка виконується крок за кроком в коротких ітераціях, називаних спринтами, кожен з яких приводить до створення функціональності, готової для випуску.

2. Співпраця та комунікація. Акцент робиться на комунікації та співпраці між учасниками проекту - клієнтами, розробниками, тестувальниками тощо.

3. Змінність вимог. Agile визнає, що вимоги можуть змінюватися протягом розробки, тому надається пріоритет індивідуальним взаємодіям та роботі над контрактами.

4. Активна взаємодія з клієнтом. Запропонована функціональність представляється клієнту для отримання зворотного зв'язку, що дозволяє швидко адаптувати продукт під його потреби.

5. Самоорганізація команд. Команди розробки самостійні та самоорганізовані, що сприяє збільшенню ефективності та творчості.

6. Регулярні засідання для оцінки та планування. Спринти завершуються регулярними оціночними та планувальними засіданнями, на яких команда оцінює прогрес та планує наступні кроки [27].

При застосуванні Agile, розробка поділяється на короткі цикли, що дозволяють швидше виправляти помилки та адаптувати продукт до змін у вимогах. Це особливо корисно в умовах швидко мінливого ринку та технологічних інновацій.

1.3 Методологія Waterfall

Методологія "Водоспад", також відома як "Модель водоспаду", представляє собою послідовний процес розробки програмного забезпечення, в

якому прогрес неухильно рухається до завершення на етапах проекту (таких як аналіз, проектування, розробка, тестування). Цей підхід передбачає повне документування проекту заздалегідь, включаючи інтерфейс користувача, історії користувача та всі варіанти та результати функцій [17].

Мета методології "Водоспад" слідує принципу "двічі виміряти, один раз відрізати". Детальне розслідування та повне дослідження особливостей продукту проводяться заздалегідь, усуваючи (більшість) ризиків проекту. Оскільки більшість досліджень вже виконано заздалегідь, оцінки часу, необхідного для кожної вимоги, є більш точними, що забезпечує більш передбачувану дату випуску.

Фази проекту для розробки програмного забезпечення за використання моделі "Водоспад":

1. Аналіз. Команда розробників продуктів аналізує вимоги та повністю розуміє проблеми.

2. Дизайн. Розробники програмного забезпечення створюють технічне рішення проблем, викладених у вимогах до продукту, включаючи сценарії, макети та моделі даних.

3. Реалізація. Технічна реалізація розпочинається після затвердження проекту. Це зазвичай є найкоротшим етапом, оскільки дослідження та проектування вже виконані заздалегідь.

4. Тестування. Після завершення реалізації необхідно провести тестування перед випуском продукту клієнтам. Команда тестування використовує проектну документацію для створення тестових прикладів [23].

Однак зміни є необхідною частиною розробки, і вони можуть виникнути у

різних формах.

В моделі "Водоспад" зміни коштують дорого через велику інвестицію у ранні етапи проекту.

Опір змінам також може виникнути через інвестиції у вимоги на ранніх етапах.

1.4 Завдання безпеки в DevOps

Забезпечення безпеки в середовищі DevOps є важливим завданням, оскільки прискорений темп розробки та впровадження може створити нові вектори загроз та підвищити ризики безпеки. Ключові завдання безпеки в DevOps:

1. Автоматизоване тестування безпеки:
   - необхідно використовувати інструменти для автоматизованого тестування безпеки, такі як OWASP ZAP, Burp Suite, або інші;
   - включати тестування на предмет вразливостей у різних етапах життєвого циклу розробки.
2. Моніторинг та аналіз безпеки:
   - забезпечує постійний моніторинг заходів безпеки у виробничому середовищі;
   - необхідно використовувати системи журналювання та аналізу безпеки для виявлення незвичайної активності.
3. Інтеграція безпеки в CI/CD:
   - здійснює автоматизоване тестування безпеки в процесі неперервної інтеграції та доставки (CI/CD);
   - впроваджує засоби контролю якості коду, які виявляють потенційні проблеми безпеки.
4. Менеджмент доступу і привілеїв:
   - використовує концепції найменшого спільного забезпечення (Least Privilege) для обмеження доступу до ресурсів;

- переглядає та оновлюйте права доступу регулярно.

5. Контроль конфіденційності та цілісності:
   - зашифровує конфіденційні дані у спокійному та транзитному стані;
   - встановлює механізми перевірки цілісності для захисту від змін вірусів.
6. Захист інфраструктури як коду:
   - включає принципи безпеки в інфраструктуру як коду (Infrastructure as Code
   - IaC);
   - використовує інструменти, які дозволяють автоматизувати та контролювати інфраструктурні зміни.
   - Безпека залежностей:
   - перевіряє безпекові оновлення та вразливості в залежностях;
   - використовує механізми контролю версій для слідкування за використанням безпечних версій пакетів.
8. Навчання та освіта з безпеки:
   - забезпечує регулярне навчання команди розробки та IT-персоналу з питань безпеки;
   - заохочує використання найкращих практик та стандартів безпеки [35].

Забезпечення безпеки повинно бути інтегровано в усі етапи розробки та використовувати принципи "безпека з самого початку" (Security by Design).

1.5 Архітектура середовища безпечної розробки різновиду DevSecOps

DevSecOps еволюціонував з DevOps, коли розробницькі команди зрозуміли, що модель DevOps не вирішує достатньо проблем безпеки. Натомість DevSecOps виник як спроба інтегрувати управління безпекою на етапі розробки, а не модернізувати систему безпеки на етапі збірки.

Цей підхід дозволяє почати процес забезпечення безпеки програми в самому початку розробки, а не в кінці розробки.

DevSecOps наголошує, що розробники повинні створювати код з урахуванням безпеки та ставить перед собою завдання вирішити проблеми

безпеки, які можливо не вирішує DevOps.

Безпека розпочинається там, де починається цикл DevOps - з доступу до репозиторіїв та джерел коду [28].

В умовах швидкості та гнучкості можуть виникати ризики безпеки, оскільки код може бути недостатньо перевіреним або, що ще гірше, його можуть випустити в продакшн без відповідної політики безпеки для збереження часу.

Коли розробники вилучають код з репозиторію та починають над ним працювати, на певному етапі його потрібно розмістити на призначеній для цього інфраструктурі. У розробці код може працювати коректно, оскільки він ще не взаємодіє з виробничими системами. Але при підготовці до релізу виробництву йому потрібно буде встановити з'єднання.

На багатьох підприємствах практика полягає в тому, що безпека є останнім етапом перед остаточним розгортанням і не вбудована в DevOps-цикл. Проте це може збільшити поверхню атаки на код і систему.

Підсумовуючи, безпека повинна бути вбудована в DevOps як спільна відповідальність розробників, операторів та інженерів безпеки. Це повинно відбуватися на кожному етапі циклу розробки та розгортання, від витягування коду з репозиторію до фактичного коміту коду та відправлення його виробництву [24].

DevSecOps включає три основні рівні:

1. Культура:
   - DevSecOps покладає акцент на культуру, що визнає відповідальність кожного члена команди за безпеку;
   - це не лише технічний рівень, і важливо, щоб всі учасники команди діяли з урахуванням аспектів безпеки;
   - наявність інженера з безпеки в команді може допомогти забезпечити відповідність стандартам та політикам безпеки.
2. Безпека за дизайном:
   - безпека вбудована на всіх рівнях системи;

- включає визначену архітектуру, що охоплює всі аспекти безпеки, такі як аутентифікація, авторизація, конфіденційність, цілісність даних, доступність, а також виправлення та коригувальні дії під час атак.

3. Автоматизація:
   - у DevSecOps наголошується на автоматизації, включаючи безпеку;
   - мета автоматизації полягає в уникненні людських помилок та виявленні потенційних проблем безпеки під час розробки;
   - автоматизовані репозиторії можуть виконувати сканування коду для виявлення вразливостей або невідповідних компонентів.

Ці заходи спрямовані на створення ефективної системи безпеки для модулів та контейнерів, що використовують різні рівні контролю доступу та аудиту для забезпечення безпеки та стабільності системи.

Висновки до першого розділу

З інтенсивним розвитком технологій програмне забезпечення стає все більш складним, що призводить до виникнення нових методологій розробки. Для забезпечення якості на протязі всього циклу розробки важливо використовувати гнучкі методології розробки. У даному випадку були розглянуті та порівняні такі методології, як DevOps, Agile та Waterfall.

1. DevOps:
   - основні переваги: висока автоматизація, швидке постачання та розгортання продуктів, оперативне вирішення проблем протягом усього циклу розробки;
   - практики: постійний розвиток, неперервне тестування, неперервна інтеграція, неперервна доставка, неперервне розгортання, неперервний моніторинг та інфраструктура як код.
2. Agile:

- основні принципи: акцент на людях та взаємодії, робочий продукт важливіший за документацію, співпраця з клієнтом важливіша за умови контракту, готовність до змін важливіша за виконання плану;
- підходи: Scrum, Kanban, XP (Extreme Programming).

3. Waterfall:
   - основні етапи: аналіз, дизайн, реалізація, тестування;
   - принцип: послідовний процес розробки, детальне планування та документування перед реалізацією [33].

Кожна з цих методологій має свої переваги та особливості. DevOps вигідний завдяки автоматизації та швидкому розгортанню, Agile наголошує на гнучкості та співпраці, тоді як Waterfall пропонує послідовний та докладний підхід. Вибір методології залежить від конкретних потреб та характеристик проекту.

РОЗДІЛ 2
РОЗРОБКА МЕТОДИКИ РИЗИК-ОРІЄНТОВАНОГО ТЕСТУВАННЯ

2.1 Вимоги до процесу тестування в DevOps

Процес тестування в DevOps включає в себе важливі етапи, щоб забезпечити високу якість програмного продукту та ефективне взаємодію між розробкою та операціями. Ось деякі вимоги до процесу тестування в рамках DevOps:

1. Автоматизація тестування:
   - розробка та впровадження автоматизованих тестів для швидкої та ефективної перевірки функціональності, продуктивності, безпеки та інших характеристик;
   - інтеграція автоматизованих тестів у процес неперервної інтеграції/неперервної доставки (CI/CD).
2. Неперервна інтеграція/неперервна доставка (CI/CD):
   - включення тестів в процес CI/CD для автоматичного тестування коду при кожній зміні;
   - забезпечення взаємодії між розробниками та тестувальниками під час CI/CD процесу.
3. Інфраструктура як код (IaC):
   - використання IaC для автоматизації створення, конфігурування та управління тестовими середовищами;
   - забезпечення консистентності тестових середовищ в усіх етапах життєвого циклу розробки.
4. Моніторинг та логування:
   - розробка та використання механізмів моніторингу для виявлення проблем під час тестування та вирішення їх;

- логування та аналіз результатів тестів для швидкого виявлення та виправлення дефектів.

5. Системи керування версіями:
   - зберігання тестових скриптів та конфігураційних файлів у системах керування версіями для ефективного спільного використання та відстеження змін.
6. Тестування безпеки:
   - включення тестів безпеки в процес тестування для виявлення та усунення потенційних вразливостей програмного продукту.
7. Комунікація та співпраця:
   - забезпечення відкритої та ефективної комунікації між розробниками, тестувальниками та операторами;
   - спільна робота над вирішенням виявлених проблем та удосконаленням процесу тестування.

Ці вимоги допомагають створити інтегрований та автоматизований процес тестування в межах DevOps, щоб забезпечити високу якість програмного продукту та швидке впровадження змін.

2.2 Ризик-орієнтований підхід до тестування

Ризик-орієнтований підхід до тестування — це стратегія тестування, спрямована на виявлення та зменшення ризиків, пов'язаних з розробкою та впровадженням програмного продукту. Основна ідея полягає в тому, щоб найкраще використовувати обмежені ресурси для тестування та фокусувати їх на тих частинах системи, які мають найвищий ризик неуспіху або найбільший вплив на бізнес-процеси [40].

Ризик-орієнтований підхід допомагає ефективно використовувати ресурси, скорочує час тестування та підвищує ймовірність виявлення та усунення критичних проблем. Це особливо корисно в умовах розвитку проектів з великою динамікою та обмеженими ресурсами.

Тестування на основі ризику (RBT) – це підхід до тестування програмного забезпечення, що заснований на врахуванні ймовірності виникнення ризиків.

Цей метод включає в себе оцінку ризиків, основану на складності програмного забезпечення, його критичності для бізнесу, частоті використання, областях, де може виникнути дефект і т.д. Тестування на основі ризику пріоритезує виконання тестів, враховуючи особливості та функції програмного продукту, які є найбільш ефективними [39].

Цей підхід до тестування на основі ризиків грає важливу роль у вдосконаленні зусиль забезпечення якості та максимізації користі від тестування при мінімізації ризику для бізнесу. Основні завдання цього методу включають:

1. Дозволяти перевірити "що хочуть клієнти" з точки зору бізнесу.
2. Виконувати графік роботи з очікуваною якістю.
3. Оптимізувати зусилля забезпечення якості.

Ризик-орієнтоване тестування застосовується в таких випадках:

- коли є обмеження часу, вартості та ресурсів проекту і необхідно оптимізувати використання ресурсів;
- коли програма складна та використовує нові технології, пов'язана з численними проблемами;
- коли програма є науково-дослідним проектом, має велику кількість невідомих та ризиків.

Нижче подано послідовні етапи ризик-орієнтованого підходу:

- аналіз вимог до продукту;
- перевірка та огляд документації (SRS, FRS, Usecases);
- визначення ризиків;
- оцінка ризиків за однією з методик оцінки ризиків;
- визначення загального рейтингу ризиків;
- використання реєстру ризиків для перерахунку списку ідентифікованих ризиків. Регулярне оновлення та відстеження ризиків через час;

- розподіл пріоритетних вимог на основі рейтингу;
- визначення процесу тестування на основі ризику;
- розгляд критичних, високих та середніх ризиків для планування, тестування та моніторингу прогресу. Низькі ризики можна зберігати в списку спостереження;
- планування та визначення тестів відповідно до рейтингу;
- застосування відповідного підходу до тестування та методів проектування тестів для охоплення елементів з найвищим рівнем ризику;
- розробка тестів з урахуванням кількох функціональних можливостей та наскрізних бізнес-сценаріїв;
- створення плану тестування;
- виконання тестів;
- відстеження зв'язку між елементами ризику, тестами, результатами тестів та виявленими дефектами під час тестування. Всі виконані стратегії тестування повинні зменшити ризики якості;
- оцінка критеріїв завершення. Повне тестування зон високого ризику, залишкові ризики зведені до мінімуму;
- звіти про результати тестування на основі ризику та аналіз показників;
- переоцінка існуючих та нових ризиків;
- оновлення реєстру ризиків;
- розробка планів надзвичайних ситуацій;
- аналіз дефектів та профілактика для усунення дефектів;
- повторне та регресійне тестування для підтвердження виправлення дефектів на основі попереднього аналізу ризику;
- оцінка залишкового ризику;
- моніторинг і контроль ризиків. [24]

За допомогою цих кроків можна створити блок-схему етапів проведення ризик-орієнтованого тестування, яка зображена на рис. 2.1.

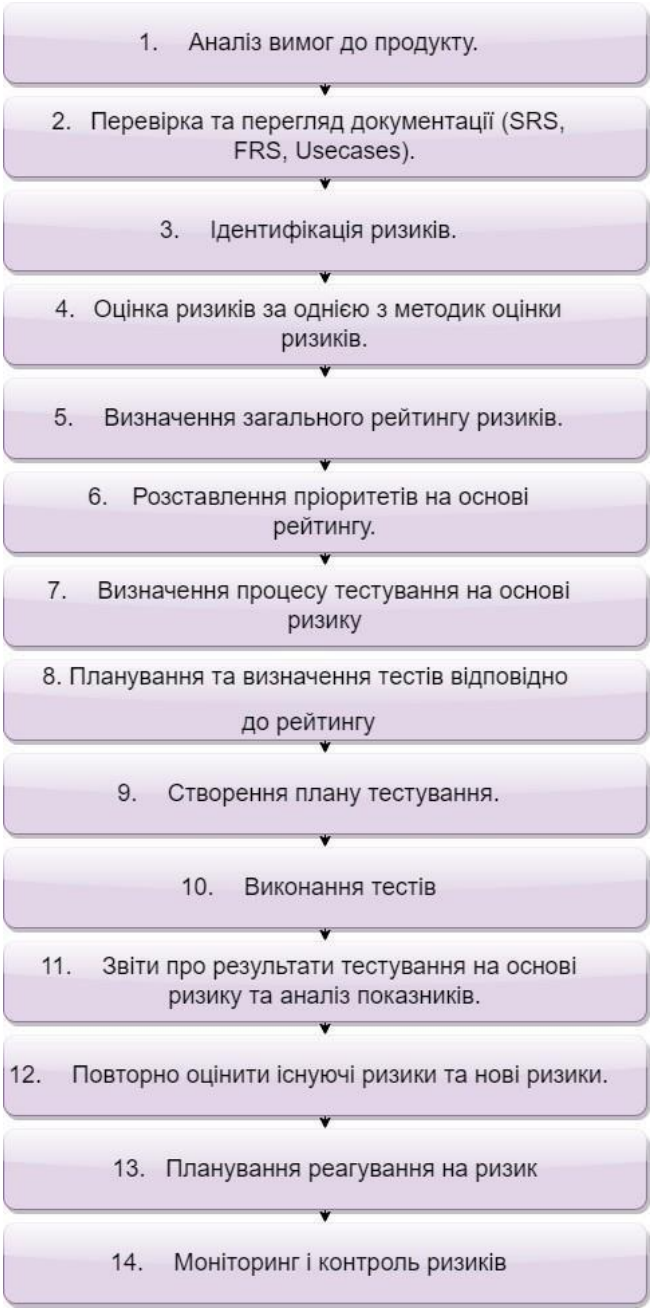


Рис. 2.1 – Етапи проведення тестування

2.3 Вибір методів оцінки ризиків

Оскільки в управлінні ризиків є етап аналізу та оцінки ризиків, доцільним буде розглянути різні методики оцінки та аналізу ризиків для вибору оптимальної.

При виборі методів оцінки ризиків для тестування на основі ризиків (RBT), існує кілька підходів та інструментів. Декілька основних методів включають:

1. Аналіз важливості функцій (FMEA - Failure Mode and Effect Analysis):
   - Оцінюється важливість кожної функції програмного продукту та визначаються можливі ризики для кожної з них;
   - визначаються ймовірність виникнення ризиків та їх вплив на бізнес- процеси.
2. Матриця ймовірностей та впливу:
   - ризики оцінюються за ймовірністю виникнення та потенційним впливом;
   - створюється матриця, в якій кожному ризику присвоюються бали відповідно до ймовірності та впливу.
3. Дерево ризиків:
   - ризики моделюються у вигляді дерева, де кожен вузол представляє конкретний ризик, ймовірність виникнення та його вплив;
   - дозволяє визначити кумулятивний ефект ризиків та їх вплив на проект.
4. Експертні оцінки:
   - залучення експертів для оцінки ризиків;
   - експертні оцінки можуть використовуватися для визначення ймовірності виникнення ризиків та їх впливу.
5. Сценарійна аналітика:
   - розробка різних сценаріїв для визначення можливих ризиків та їхніх наслідків;
   - дозволяє виявити ризики, які можуть виникнути в різних умовах [38].

Обираючи метод оцінки ризиків, важливо враховувати конкретні вимоги та характеристики проекту, а також доступні ресурси. Комбінування декількох методів може також забезпечити більш комплексний погляд на ризики та їх управління.

Щоб переконатись в коректності управління ризиками був проведений експеримент для виявлення кращого аналізу та оцінки ризиків. В ході дослідження були взяті наступні методики – DREAD Microsoft, PrisMA, PRAM, FMEA та FTA.

FMEA (Failure Mode and Effects Analysis) – це формальна методика, що використовується для систематичного виявлення та оцінки можливих відмов (Failure Modes), їхніх можливих наслідків (Effects) та ймовірності виникнення таких відмов. Цей метод дозволяє здійснювати прогнозування та управління ризиками у процесах розробки, виробництва та експлуатації продукції.

Основні етапи FMEA включають:

1. Ідентифікація відмов (Failure Modes). Визначення всіх можливих способів відмови компонента, системи чи процесу.
2. Оцінка ймовірності виникнення відмови (Occurrence). Оцінка ймовірності того, що конкретна відмова станеться.
3. Оцінка виявлення відмови (Detection). Оцінка ймовірності виявлення відмови до того, як вона може спричинити серйозні наслідки.
4. Оцінка серйозності наслідків відмови (Severity). Визначення серйозності або важливості наслідків відмови.
5. Розрахунок ризику (Risk Priority Number, RPN). Множення оцінок ймовірності, виявлення та серйозності для кожної відмови для отримання числового показника ризику.
6. Розробка заходів з управління ризиками (Risk Mitigation). Розробка та впровадження заходів для зменшення ймовірності виникнення відмов, поліпшення виявлення відмов або зменшення серйозності наслідків.
7. Повторення аналізу (Reassessment). Періодичне переглядання та оновлення FMEA для врахування змін у процесі, системі або продукті [34].

FMEA є важливим інструментом для покращення якості, надійності та безпеки продуктів та процесів, а також для управління ризиками на різних етапах їхнього життєвого циклу.

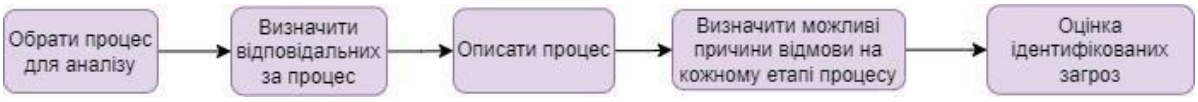


Рис. 2.2 – Блок-схема для аналізу та оцінки ризиків методикою FMEA

Метою FMEA є впровадження заходів для усунення або зменшення відмов, починаючи з найбільш пріоритетних.

Аналіз режимів і наслідків відмов також фіксує поточні знання та заходи щодо ризиків відмов для використання у процесі постійного вдосконалення. FMEA використовується під час проектування для попередження відмов. Пізніше його застосовують для контролю під час етапів до та під час поточної експлуатації процесу. В ідеалі FMEA починається на ранніх концептуальних етапах проектування і здійснюється протягом усього терміну служби продукту чи послуги.

PRISMA (Product RISk Management) - це підхід для визначення сфер, які є найважливішими для тестування, тобто визначення областей, які мають найвищий рівень ділового та технічного ризику. Етапи представлені на рис. 2.3.

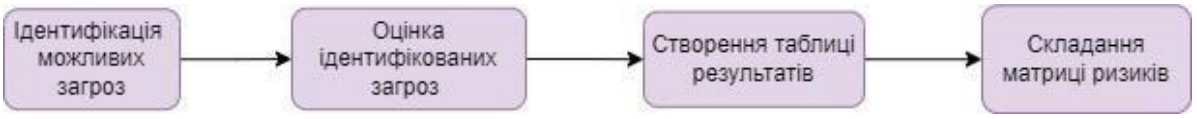


Рис. 2.3 – Блок-схема для аналізу та оцінки методикою PRISMA

В процесі PRISMA ключовим елементом є розробка матриці ризику. Для кожного ідентифікованого ризику визначаються його можливий вплив та ймовірність настання небажаних подій. Після надання числових значень впливу та ймовірності ризик розміщується у матриці ризику продукту.

Стандартна матриця ризику розділяється на чотири зони, кожна з яких представляє різний рівень та тип ризику. Кожен рівень також вимагає використання відмінного підходу до тестування, який слід задокументувати в (основному) плані тестування [16]. Таким чином, матриця ризику продукту може слугувати основою для всіх тестів, проведених у рамках проекту [15].

PRAM охоплює процеси, прийоми та методи, що дозволяють проводити аналіз та управління ризиками, пов'язаними з проектом. Етапи оцінки представлені на рис. 2.4.

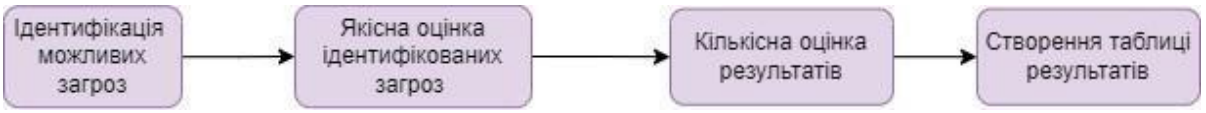


Рис. 2.4 – Блок-схема для аналізу та оцінки методикою PRAM

Розглянемо дві основні частини методики PRAM:

1. Якісний аналіз ризиків:
   - ідентифікація ризиків. Визначення можливих подій або умов, які можуть вплинути на успішне виконання проекту;
   - суб'єктивна оцінка ризиків. Оцінка ймовірності та впливу ризиків за допомогою суб'єктивних методів, таких як експертні оцінки або анкетування учасників проекту.
2. Кількісний аналіз ризиків:
   - об'єктивна оцінка ризиків. Використання числових методів для кількісної оцінки ймовірності та впливу ризиків. Це може включати в себе використання статистичних моделей, симуляцій або інших кількісних інструментів.

Методика PRAM допомагає командам проектів і організаціям розпізнавати, оцінювати та керувати ризиками, що можуть виникнути під час виконання проектів. Це важливий етап управління проектом, оскільки дозволяє передбачити та підготуватися до можливих труднощів та непередбачуваних обставин [7].

DREAD - це методика аналізу ризиків, розроблена для оцінки рівня загроз в інформаційній безпеці та програмуванні. Абревіатура DREAD представляє собою п'ять аспектів, які оцінюються для кожної загрози:

1. Damage (Збитки). Які збитки може завдати атака? Це може включати фінансові втрати, втрати даних, втрату репутації та інше.
2. Reproducibility (Відтворюваність). Наскільки легко можливо відтворити атаку? Якщо атака легко відтворюється, то загроза вважається більш серйозною.
3. Exploitability (Використовуваність). Наскільки складно використати вразливість або вразливості для здійснення атаки?
4. Affected users (Кількість користувачів, яких стосується). Скільки користувачів може бути потенційно постраждалими від атаки?

5. Discoverability (Виявленість). Наскільки легко можна виявити цю вразливість чи загрозу? Якщо вона легко виявляється, то це може викликати більш великий ризик.

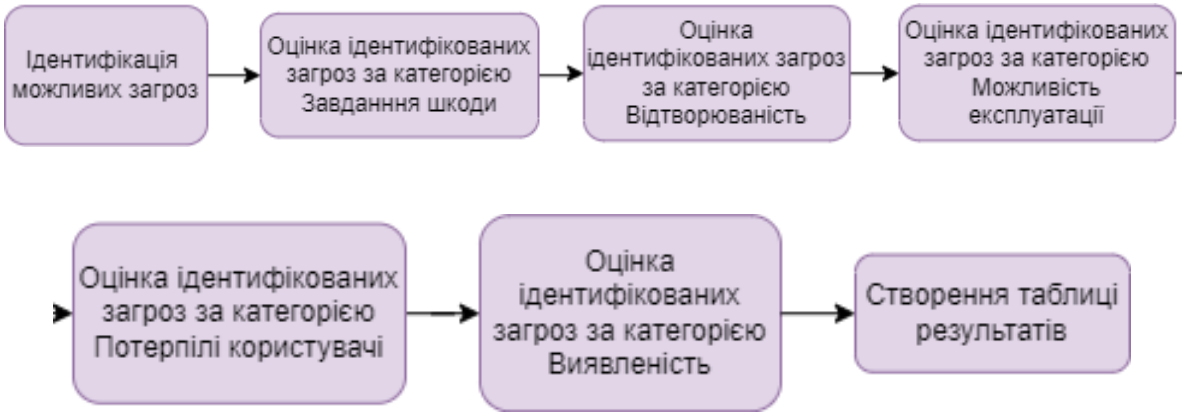


Рис. 2.5 – Блок-схема для аналізу та оцінки методикою DREAD

Кожен аспект оцінюється на числову шкалу, і обчислюється сумарний бал DREAD для кожної загрози. Ця система допомагає приорітезувати ризики та визначати стратегії для їх управління в інформаційних та програмних системах.

Для проведення експерименту був взятий конкретний перелік ризиків, та була проведена оцінка кожним з методів. Методологія DevOps накладає певні обмеження на вибір оптимального методу оцінки ризиків, зокрема для Azure DevOps встановлені такі типові обмеження:

– обмеження часу виконання запитів - 30 секунд;;
– результати, повернені запитом, обмежені 20 000 рядків;
– довжина запиту – не більше 32 000 символів [17].

Важливим аспектом в методології DevOps є оформлення результатів, яке повинно підвищити розуміння видів ризиків, їх можливого впливу та шляхів обробки. Обрана методика повинна забезпечити простежуваність, відтворюваність та верифікацію процесу та результатів.

Оскільки головною метою методології DevOps є розробка якісних продуктів та швидкі випуски, для порівняння методик оцінки та аналізу ризиків доцільно вибрати такі критерії, як точність оцінювання для забезпечення якості та час оцінювання для швидкості виконання.

Також на вибір методу впливатимуть фактори доступності ресурсів, практичний досвід, навички та можливості групи оцінки ризиків, а також доступний бюджет для зовнішніх ресурсів.

Додатково обрана методика оцінки та аналізу ризиків повинна бути добре інтегрованою з ризик-орієнтованим підходом до тестування. З урахуванням наданої інформації та вимог до тестування, результати експерименту будуть порівнюватися за такими критеріями, як витрати ресурсів, час оцінювання, точність оцінювання та інтеграція з RBT. Отримані результати представлені в таблиці 2.1.

Таблиця 2.1

Результати порівняння

Методика	**Витрати ресурсів**	**К-сть атрибутів для оцінки**	**Час оцінювання**	**Точність оцінювання**	**Інтегація з RBT**
PRisMA	Середні	2	Виконується швидко	Має середньоточність	Відмінна
PRAM	Середні	2	Виконується швидко	Має середньоточність	Відмінна
FMEA	Середні	3	Середні витрати	Висока точність	Відмінна
FTA	Вище середніх	2	Потребує часу	Висока точність	Середня
DREAD	Вище середніх	5	Потребує багато часу	Високаточність	Середня

Під час проведення досліджень було встановлено, що методика оцінки ризиків DREAD виходить на лідерські позиції за всіма показниками, у той час як FMEA та FTA знаходяться на другому місці. Однак, враховуючи вимоги методології розробки DevOps до оперативного управління ризиками, виявлено, що методика DREAD не є найкращим вибором через свою недостатню швидкість.

З цього приводу пропонується використовувати методику FMEA, яка продемонструвала кращі результати і, водночас, має менше атрибутів для оцінки, що призводить до заощадження часу.

Таким чином, методика FMEA вважається оптимальною з обраної перспективи, враховуючи вказані критерії. [19]

Методика FMEA (Failure Mode and Effect Analysis) є інструментом для ідентифікації потенційних ризиків і небезпек у процесах, системах або продуктах з метою їх подальшого управління. Застосування FMEA в аспекті кібербезпеки може допомогти визначити і оцінити можливі порушення безпеки і їхні можливі впливи на організацію. Нижче наведено кроки для застосування методики FMEA в контексті кібербезпеки:

1. Визначення об'єкта дослідження: ідентифікація систем, мереж, програм або інших елементів, які підлягатимуть аналізу в контексті кібербезпеки.
2. Визначення відмов: визначення потенційних відмов або порушень безпеки, які можуть статися в обраному об'єкті.
3. Оцінка ідентифікованих відмов: визначення ймовірності виникнення кожної відмови; оцінка серйозності (впливу) кожної відмови на безпеку.
4. Визначення виявлення відмов: визначення того, як швидко або ефективно можна виявити кожну відмову.
5. Підрахунок ризику: розрахунок ризику для кожної відмови, який враховує ймовірність, серйозність та ефективність виявлення.
6. Розробка стратегій управління ризиками: розроблення стратегій для зменшення ймовірності виникнення відмов, зменшення їхнього впливу або покращення виявлення.
7. Впровадження заходів управління ризиками: реалізація визначених стратегій та заходів для кожної ідентифікованої відмови.
8. Постійний моніторинг та оновлення: регулярний моніторинг ефективності впроваджених заходів та оновлення FMEA з урахуванням нових загроз і змін в системі [35].

Застосування методики FMEA в аспекті кібербезпеки дозволяє організаціям ідентифікувати, оцінювати та управляти ризиками, пов'язаними із можливими відмовами в кібербезпеці, тим самим покращуючи загальний рівень безпеки і витрат.

2.4 Методика ризик-орієнтованого тестування в конвеєрі DevSecOps

Ризик-орієнтоване тестування є важливою частиною конвеєра DevSecOps, оскільки дозволяє ідентифікувати та вирішувати проблеми з безпекою на ранніх етапах розробки. Нижче подано основні етапи і підходи до ризик-орієнтованого тестування в конвеєрі DevSecOps:

1. Ідентифікація загроз та ризиків:
   - аналіз ідентифікованих загроз і ризиків в ініційальній фазі розробки;
   - використання методів, таких як аналіз загроз (threat modeling) для визначення потенційних атак та слабких місць в системі.
2. Визначення тестових сценаріїв:
   - розробка тестових сценаріїв, які охоплюють ідентифіковані ризики та потенційні вразливості;
   - врахування тестування функціональності, витоків інформації, аутентифікації, авторизації та інших аспектів безпеки.
3. Автоматизація тестів:
   - використання інструментів для автоматизації безпекових тестів для швидкого та ефективного виявлення вразливостей;
   - інтеграція цих тестів в конвеєр DevSecOps для автоматизованої перевірки безпеки на різних етапах розробки.
4. Використання засобів статичного та динамічного аналізу коду:
   - застосування засобів статичного аналізу коду для виявлення потенційних проблем безпеки на етапі написання коду;
   - використання динамічного аналізу для тестування додатку в реальному часі під час його виконання.

5. Інтеграція відмовостійкості:
   - розробка тестів для відмовостійкості, щоб перевірити, як система реагує на атаки та відмови;
   - включення тестів з використанням інструментів, які моделюють атаки, щоб визначити, наскільки ефективно система реагує на загрози.
6. Моніторинг безпеки в реальному часі:
   - використання систем моніторингу для виявлення потенційних атак або відхилень в роботі системи;
   - інтеграція моніторингу безпеки в конвеєр DevSecOps для негайної реакції на події, пов'язані з безпекою.
7. Навчання та удосконалення:
   - аналіз результатів тестування для виявлення слабких місць та вразливостей;
   - використання цієї інформації для постійного удосконалення процесів розробки та безпеки [10].

Ризик-орієнтоване тестування в конвеєрі DevSecOps допомагає забезпечити, що аспекти безпеки враховуються на різних етапах розробки та що потенційні загрози виявляються та вирішуються на ранніх етапах розробки, забезпечуючи високий рівень безпеки продукту.

Розглянемо проведення ризик-орієнтованого тестування на кожному етапі в послідовності. На етапі Збірки необхідно виконати наступні кроки:

1. Визначення ризиків.
2. Аналіз та оцінка ризиків.
3. Створення загального рейтингу ризиків.
4. Призначення пріоритетів вимог на основі рейтингу.
5. Визначення процесу тестування, базуючись на ризиках.
6. Планування та визначення тестів згідно з рейтингом.

В цьому контексті доцільно провести тестування, включаючи такі тести як:

- перевірка залежностей в коді;
- статичний аналіз.

Під час статичного аналізу інструменти переконуються, що код повний та послідовний, перевіряючи відповідність політик безпеки та галузевих стандартів. Такий огляд повинен забезпечити детальну оцінку якості коду та відповідність документації [23].

Залежно від рейтингу ризиків, тести повинні враховувати найвищий рівень ризиків. Важливо зауважити, що техніки тестування можна застосовувати паралельно для відповідності вимогам методології DevOps.

1. Створення плану тестування.
2. Виконання тестів.
3. Оцінка критеріїв виходу, забезпечення повного покриття зон високого ризику та вирішення залишкових ризиків.
4. Звіти про результати тестування, аналіз ризиків та показників.

На етапі Тестування повторюються аналогічні кроки:

1. Визначення ризиків.
2. Аналіз та оцінка ризиків.
3. Створення загального рейтингу ризиків.
4. Призначення пріоритетів вимог на основі рейтингу.
5. Визначення процесу тестування, базуючись на ризиках.
6. Планування та визначення тестів згідно з рейтингом. Тести на цьому етапі можуть включати:

– перевірка залежностей в пакетах для контейнерів;

– динамічний аналіз, включаючи автоматизовані техніки тестування безпеки веб-додатків, сканування безпеки API та тестування безпеки на основі поведінки [42].

Подальші кроки включають створення плану тестування, виконання тестів, оцінку , а також оновлення реєстру ризиків та планів для надзвичайних ситуацій.

На етапі Розгортання повторюються аналогічні кроки, де також рекомендується провести тести безпеки веб-додатків, сканування безпеки API та тестування безпеки на основі поведінки.

Етапи Експлуатації та Моніторингу покладають основний акцент на WAST тестуванні, а також можливому виконанні сканування безпеки API та тестування безпеки на основі поведінки. Після кожного тестування важливо оформляти результати для подальшої оцінки та управління ризиками.

Рисунок 2.6 представляє схему різних стадій DevSecOps пайплайну, з фокусом на паралельному виконанні тестів різних методів тестування безпеки.

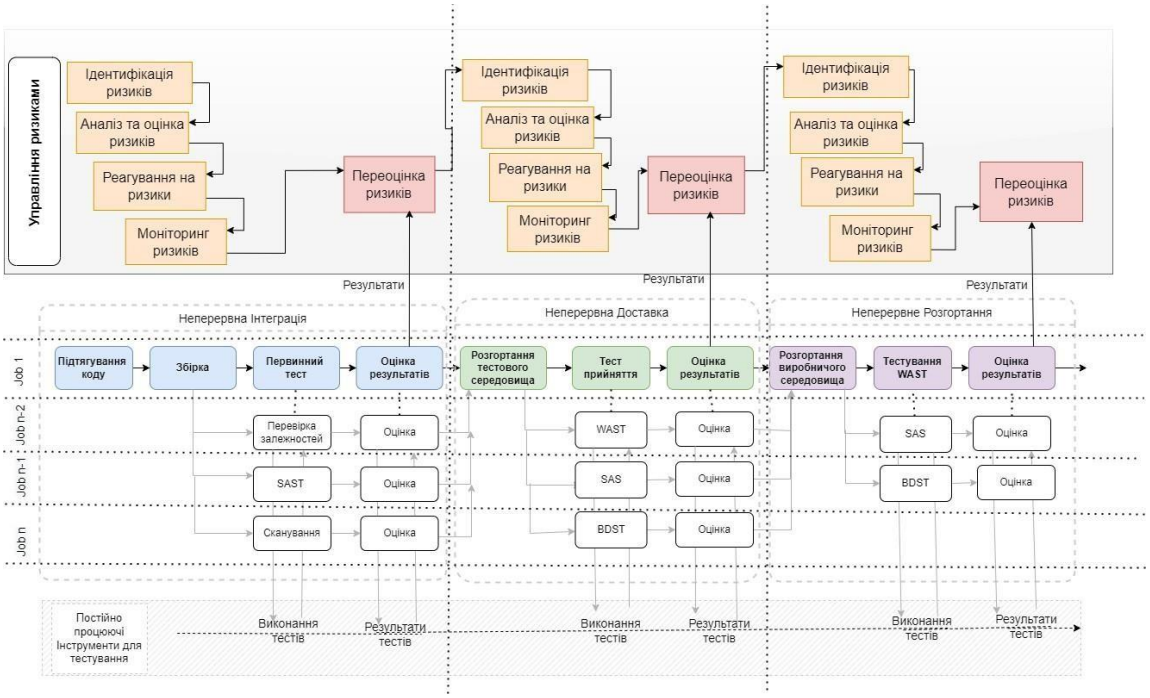


Рис. 2.6 - Схема ризик-орієнтованого тестування на різних етапах CI/CD пайплайну з паралельним виконанням тестів, з різними техніками тестування

2.5 Моделювання управління ризиками з використанням розробленої методики

Моделювання управління ризиками з використанням DevSecOps може включати в себе такі етапи:

1. Оцінка ризиків:
   - визначення потенційних загроз для безпеки програмного забезпечення;
   - оцінка вразливостей, які можуть бути використані атаками;
   - аналіз впливу можливих атак на систему та бізнес-процеси.
2. Інтеграція безпеки в CI/CD (Continuous Integration/Continuous Deployment):

- забезпечення використання інструментів автоматизації тестування на вразливості в ході процесу CI/CD;
- автоматизована перевірка вихідного коду на вразливості перед його об'єднанням у головну гілку (main/trunk);
- впровадження механізмів раннього виявлення і усунення безпекових проблем.

3. Автоматизована інтеграція інструментів безпеки:
   - використання засобів автоматичного виявлення вразливостей та аналізу коду;
   - впровадження сканерів вразливостей та інших інструментів для автоматизованого виявлення проблем безпеки.
4. Моніторинг та логування:
   - впровадження систем моніторингу безпеки для постійного виявлення аномалій та потенційних загроз;
   - логування подій та виявлення аномалій в реальному часі для швидкого реагування на можливі інциденти.
5. Культура безпеки:
   - залучення всіх учасників розробки (розробники, тестувальники, операційники) до питань безпеки;
   - проведення тренінгів та навчання з питань безпеки для всього персоналу;
   - створення культури безпеки, де всі учасники відчувають відповідальність за безпеку додатку.
6. Автоматизована реакція на інциденти:
   - розробка та впровадження автоматизованих сценаріїв реагування на інциденти безпеки;
   - постійне оновлення та вдосконалення планів реагування на інциденти з урахуванням нових загроз та вразливостей [13].

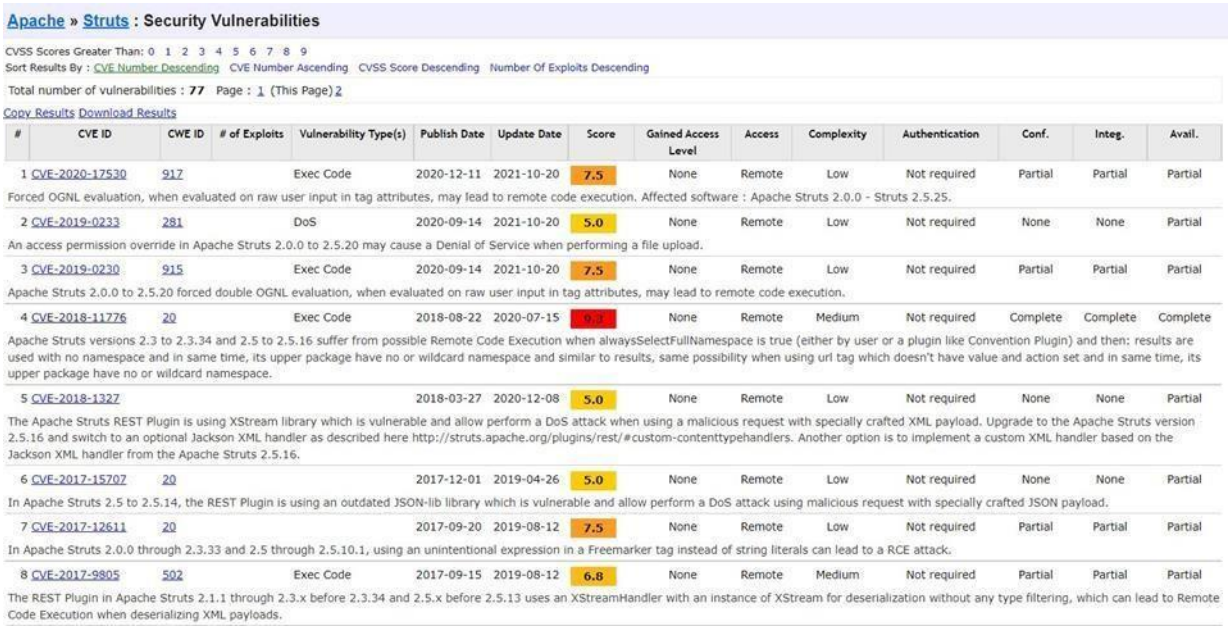
DevSecOps дозволяє розробникам і безпековим експертам працювати разом в реальному часі, спільно вирішуючи завдання безпеки та враховуючи їх в

процесі розробки. Це сприяє покращенню безпеки програмного забезпечення та ефективності відповіді на можливі інциденти.

Розглянемо управління ризиками на конкретному прикладі.

Для ілюстрації візьмемо один з ризиків, пов'язаний з продуктом - це можливість використання вразливостей, які існують у третій партійній програмній засобі (ПЗ). Перш за все, необхідно визначити застарілі бібліотеки та уразливе програмне забезпечення. Як результат, ми отримаємо перелік програм та їх версій. У нашому випадку це Apache Struts 2 версії 2.5.12. Сутність впливу від реалізації цього ризику буде досить значною. Таким чином, на основі експертної оцінки ми визначимо рівень тяжкості, який дорівнює 1 [32].

Наступним етапом буде виявлення вразливостей, яким версії компонентів піддані. У разі Apache Struts 2 існує безліч вразливостей, які можна знайти в базі даних вразливостей - CVE Details, яка представлена на рисунку 2.7.



Apache » Struts : Security Vulnerabilities

CVSS Scores Greater Than: 0 1 2 3 4 5 6 7 8 9
Sort Results By : CVE Number Descending CVE Number Ascending CVSS Score Descending Number Of Exploits Descending

Total number of vulnerabilities : 77 Page : 1 (This Page) 2

Copy Results Download Results

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score	Gained Access Level	Access	Complexity	Authentication	Conf.	Integ.	Avail.
1	CVE-2020-17530	917		Exec Code	2020-12-11	2021-10-20	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
Forced OGNL evaluation, when evaluated on raw user input in tag attributes, may lead to remote code execution. Affected software : Apache Struts 2.0.0 - Struts 2.5.25.														
2	CVE-2019-0233	281		DoS	2020-09-14	2021-10-20	5.0	None	Remote	Low	Not required	None	None	Partial
An access permission override in Apache Struts 2.0.0 to 2.5.20 may cause a Denial of Service when performing a file upload.														
3	CVE-2019-0230	915		Exec Code	2020-09-14	2021-10-20	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
Apache Struts 2.0.0 to 2.5.20 forced double OGNL evaluation, when evaluated on raw user input in tag attributes, may lead to remote code execution.														
4	CVE-2018-11776	20		Exec Code	2018-08-22	2020-07-15	6.8	None	Remote	Medium	Not required	Complete	Complete	Complete
Apache Struts versions 2.3 to 2.3.34 and 2.5 to 2.5.16 suffer from possible Remote Code Execution when alwaysSelectFullNamespace is true (either by user or a plugin like Convention Plugin) and then: results are used with no namespace and in same time, its upper package have no or wildcard namespace and similar to results, same possibility when using url tag which doesn't have value and action set and in same time, its upper package have no or wildcard namespace.														
5	CVE-2018-1327				2018-03-27	2020-12-08	5.0	None	Remote	Low	Not required	None	None	Partial
The Apache Struts REST Plugin is using XStream library which is vulnerable and allow perform a DoS attack when using a malicious request with specially crafted XML payload. Upgrade to the Apache Struts version 2.5.16 and switch to an optional Jackson XML handler as described here http://struts.apache.org/plugins/rest/#custom-contenttypehandlers. Another option is to implement a custom XML handler based on the Jackson XML handler from the Apache Struts 2.5.16.														
6	CVE-2017-15707	20			2017-12-01	2019-04-26	5.0	None	Remote	Low	Not required	None	None	Partial
In Apache Struts 2.5 to 2.5.14, the REST Plugin is using an outdated JSON-lib library which is vulnerable and allow perform a DoS attack using malicious request with specially crafted JSON payload.														
7	CVE-2017-12611	20			2017-09-20	2019-08-12	7.5	None	Remote	Low	Not required	Partial	Partial	Partial
In Apache Struts 2.0.0 through 2.3.33 and 2.5 through 2.5.10.1, using an unintentional expression in a Freemarker tag instead of string literals can lead to a RCE attack.														
8	CVE-2017-9805	502		Exec Code	2017-09-15	2019-08-12	6.8	None	Remote	Medium	Not required	Partial	Partial	Partial
The REST Plugin in Apache Struts 2.1.1 through 2.3.x before 2.3.34 and 2.5.x before 2.5.13 uses an XStreamHandler with an instance of XStream for deserialization without any type filtering, which can lead to Remote Code Execution when deserializing XML payloads.														

Рис. 2.7 – Список вразливостей для Struts2

Порахувавши фактори, ми отримуємо значення Вірогідності 2. Розрахувавши значення Виявленості отримуємо значення 2.

Значення рівня пріоритетності ризику дорівнює 4, що є Критичним і потребує зниження. Результати показані в табл. 2.2.

Таблиця 2.2

Результати ідентифікації, аналізу та оцінки ризику

Назва	Режим загрози	Вплив	Тяжкість	Вірогідність	Виявленість	RPN
Експлуатація вразливостей, що знаходяться у третьому ПЗ	Витік інформації	Несанкціонований доступ до системи	1	2	2	4

Існує можливість зменшити цей ризик шляхом обмеження кількості користувачів, які використовують функціонал, для якого потрібен зазначений компонент. Після впровадження цих змін необхідно провести повторну оцінку ризику. Враховуючи, що версія компонента залишається вразливою і експлойт може бути успішно використаний, значення тяжкості та виконуваності залишаться незмінними, але ймовірність використання експлойту буде меншою і може бути оцінена на рівні 6.

Розрахунок показника рівня ризику (RPN) за формулою

RPN = Тяжкість * Вірогідність * Виконуваність (1 * 6 * 2)

показує, що вдалося знизити рівень ризику до високого рівня (12). При подальшому плануванні реагування на ризик було рекомендовано перейти на версію Struts 2.5.17, яка вже не є вразливою до даної загрози. Після внесення цієї зміни було отримано новий результат RPN = Тяжкість * Вірогідність * Виконуваність (1 * 7 * 8), що свідчить про успішне зниження рівня ризику до середнього рівня (56).

Висновки до другого розділу

Процес тестування в DevOps включає в себе важливі етапи, щоб забезпечити високу якість програмного продукту та ефективне взаємодію між розробкою та операціями.

Застосування методики FMEA в аспекті кібербезпеки дозволяє організаціям ідентифікувати, оцінювати та управляти ризиками, пов'язаними із можливими відмовами в кібербезпеці, тим самим покращуючи загальний рівень безпеки і витрат.

Ризик-орієнтоване тестування в конвеєрі DevSecOps допомагає забезпечити, що аспекти безпеки враховуються на різних етапах розробки та що потенційні загрози виявляються та вирішуються на ранніх етапах розробки, забезпечуючи високий рівень безпеки продукту.

DevSecOps дозволяє розробникам і безпековим експертам працювати разом в реальному часі, спільно вирішуючи завдання безпеки та враховуючи їх в процесі розробки. Це сприяє покращенню безпеки програмного забезпечення та ефективності відповіді на можливі інциденти.

РОЗДІЛ 3
РЕЗУЛЬТАТИ ТЕСТУВАННЯ ТА ОЦІНКА РИЗИКІВ

3.1 Створення програмного комплексу для оцінки ризиків

Створення програмного комплексу для оцінки ризиків - це завдання, яке вимагає детального планування і впровадження кількох ключових етапів. Цей підхід може бути адаптований в залежності від конкретних вимог і умов вашого проекту.

В роботі ми застосовуємо програмний комплекс для оцінки ризиків за допомогою методики FMEA.

Аналіз режимів і наслідків збоїв програмного забезпечення (FMEA) — це структурований підхід, спрямований на виявлення й оцінку потенційних режимів збою в системі програмного забезпечення, оцінку їх наслідків і визначення пріоритетів на основі серйозності, ймовірності виникнення та можливості виявлення. FMEA виник у галузі машинобудування та спочатку використовувався у виробничих процесах. Однак його застосування було розширено до розробки програмного забезпечення, визнаючи необхідність передбачати та зменшувати ризики, пов’язані з програмним забезпеченням.

Програмний комплекс для оцінки ризиків за допомогою методики FMEA реалізовано на мові програмування AL в середовищі розробки Microsoft Business Central 14 CU 29, під управлінням операційної системи Microsoft Windows 10.

Dynamics 365 Business Central (раніше відомий як Dynamics NAV, а до цього – Navision!) – це єдине, універсальне рішення для управління всіма бізнес-процесами, а також складова бізнес-додатків Microsoft Dynamics 365. Воно також надзвичайно гнучке і може бути налаштоване згідно з будь-якими потребами бізнесу [33].

Business Central – це надійне, перевірене рішення у сфері ERP, обслуговує 160 000 клієнтів і щороку покращується компанією Microsoft. Воно має широкий

функціонал і може бути розгорнуте в загальнодоступному хмаровому середовищі, приватній хмарі або локально.

Були встановлені специфікації для серверної та клієнтської частин. Вимоги до серверної частини включають:

1. Операційні системи сервера:
   - Windows 10 Pro, Enterprise або Education (64-бітна версія);
   - Windows Server 2019 (Центр обробки даних або Стандартний);
   - Windows Server версії 1809 або новішої, Windows Server 2016 Standard, Essentials або Datacenter.
2. Веб-сервер:
   - Internet Information Server 10;
   - Internet Information Server 8.5;
   - Internet Information Server 8.0.
3. SQL Сервер:
   - Microsoft SQL Server 2019 Express, Standard, або Enterprise;
   - Microsoft SQL Server 2017 Express, Standard, або Enterprise;
   - Microsoft SQL Server 2016 Express, Standard, або Enterprise;
   - Azure SQL Database Managed Instance, Elastic Pool, або Single Database.

Щодо клієнтської частини, чи точніше, рекомендацій для використання Business Central у браузері, вказано наступне:

1. Підтримувані браузери:
   - New Microsoft Edge для Windows;
   - Google Chrome для Windows (версія 77.0 чи пізніша);
   - Mozilla Firefox для Windows (версія 69.0 чи пізніша);
   - Safari для macOS (версія 12.0 чи пізніша);
   - Internet Explorer 11;
   - Microsoft Edge Legacy.

   Щодо мобільного додатка Business Central, вказані наступні вимоги:
2. Операційні системи:

- Windows 10S, Home, Pro, Enterprise, або Education (32-бітні та 64-бітні версії);
- Android (планшет і телефони): Останні три основні версії та їх оновлення;
- IOS (IPad; або Iphone): Останні три основні версії та їх оновлення;
- програмний комплекс активується шляхом запуску сервісу Microsoft Dynamics 365 Business Central 140. Після запуску сервісу необхідно перейти за адресою http://localhost:8080/BC140 та ввести особистий логін та пароль

Microsoft Edge LegacyПісля вибору вкладки "Ідентифікація та оцінка ризиків" з'являється початковий інтерфейс. Експерт з кібербезпеки вводить дані ідентифікованих

загроз. Потім фахівець визначає ресурси, процеси чи етапи процесів, що викликають цю загрозу. Наступний етап - вибір тяжкості загрози, де програмний комплекс дозволяє вводити значення від 1 до 10, з точністю до п'яти знаків після коми. Також визначається ймовірність реалізації загрози та виявлення її. Після введення даних необхідно підтвердити їх, натиснувши кнопку "Підтвердити".

Після цих кроків формується таблиця (рис. 3.1).

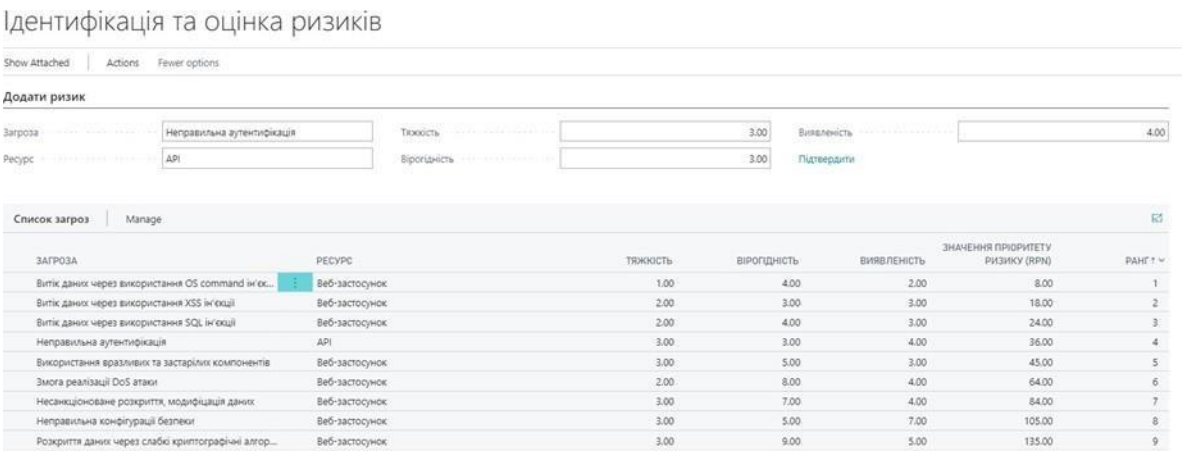


Рис. 3.1 – Формування таблиці з рангом і показником значення пріоритету ризику

Таблиця має шість колонок, кількість рядків необмежена і залежить від обсягу введених даних. Ось опис кожної колонки:

1. Загрози. В цій колонці відображаються введені загрози.

2. Ресурси. Тут відображаються введені ресурси.
3. Тяжкість загрози. Вказує обране число тяжкості, яке визначається користувачем.
4. Вірогідність. Показує обране число вірогідності, яке також вводиться користувачем.
5. Виявленість. Відображає обране число виявленості, введене користувачем.
6. Значення пріоритету ризику. У цьому полі відображається і розраховується показник ризику, обчислений за певною формулою.

$$RPN = S * O * D, \quad (3.1)$$

де S – тяжкість, O – вірогідність D – виявленість.

7. Ранг – відображається ранг загрози відповідно до значення пріоритету ризику.

Після завершення додавання всіх загроз, ресурсів і визначення показників ризику та рангів загрози фахівець з кібербезпеки обирає прийнятний ранг. Для цього він переходить у вкладку Прийняття ризиків (Рис. 3.2)

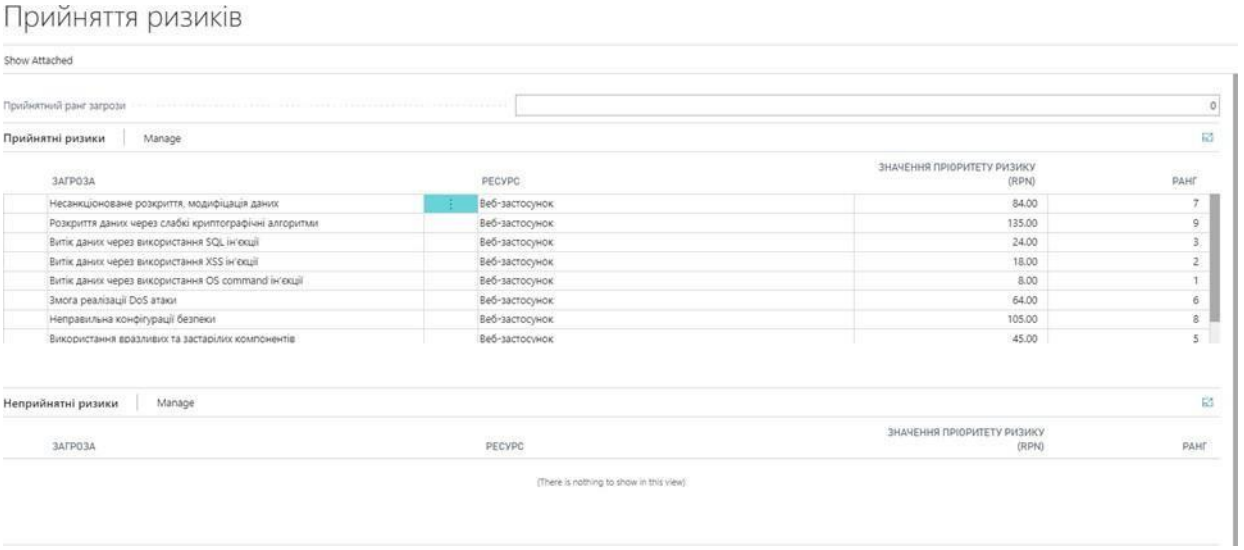


Рис. 3.2 – Інтерфейс вкладки прийняття ризиків

Після обрання прийнятного рангу загрози формується таблиця прийнятних і неприйнятних ризиків, як показано на рис. 3.3.

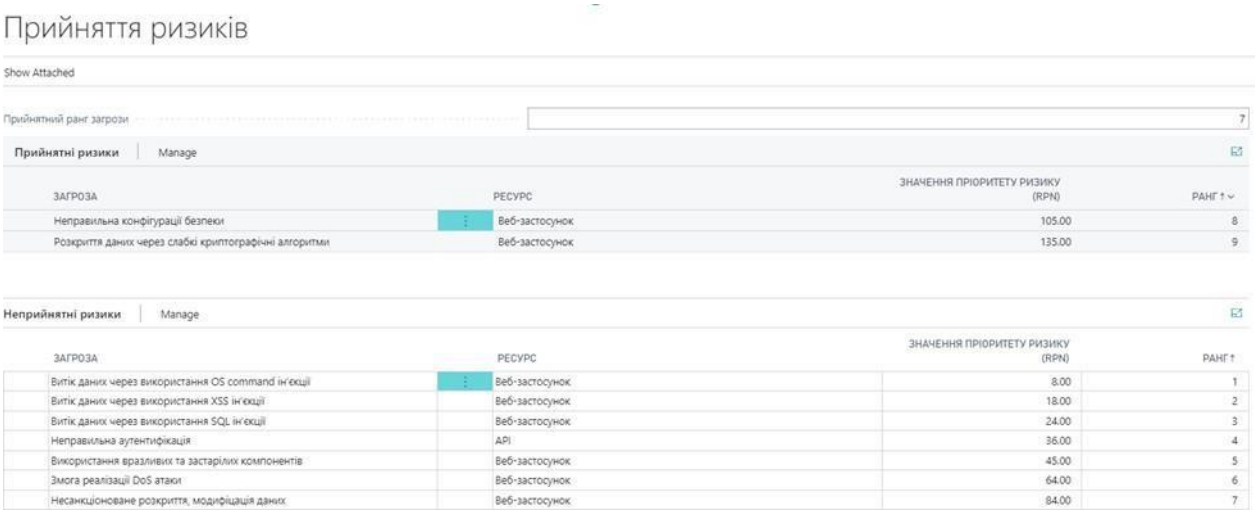


Рис. 3.3 – Приклад формування таблиці прийнятних і неприйнятних загроз

Список Неприйнятих загроз можна додати до реєстру ризиків, а список Прийнятих загроз додається до списку спостережень.

Трохи нижче на рис. 3.4 знаходиться таблиця Реагування на ризики.

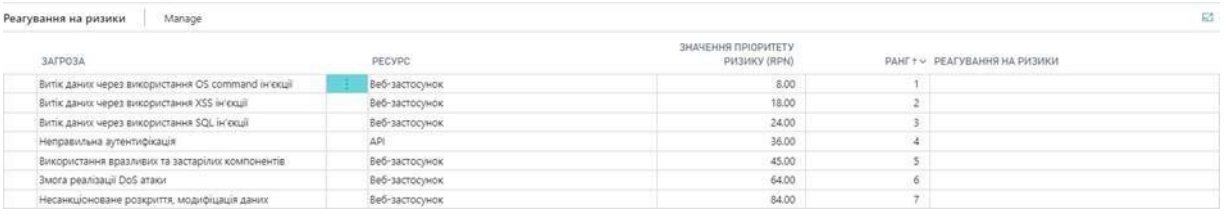


Рис. 3.4 – Інтерфейс таблиці Реагування на ризики

3.2 Результати проведення ризик-орієнтованого тестування

Ризик-орієнтоване тестування - це процес ідентифікації, аналізу та оцінки ризиків, пов'язаних із розробкою та впровадженням програмного забезпечення, та визначення тестових сценаріїв на основі цих ризиків.

Для перевірки вірності та ефективності функціонування системи управління ризиками та розробленої методики ризик-орієнтованого тестування в рамках DevSecOps, був проведений експеримент. Цей експеримент включав в себе впровадження управління ризиками та ризик-орієнтованого тестування на конвеєрі CI/CD з методології DevOps, зокрема на етапах Збірки та Тестування.

Обрано OWASP WebGoat в якості цільової програми. WebGoat — це спеціально створена програма з наміром викликати небезпеку, з одного боку, з

метою освіти, та перевірки інструментів безпеки, з іншого боку. Крім того, спільнота WebGoat надає зображення Docker WebGoat, доступне для витягування з загальнодоступного сховища. Документація WebGoat також містить інформацію про його уразливості, такі як SQLi, та способи їх використання [36].

На рис. 3.5 ілюстрація процесу запуску WebGoat.

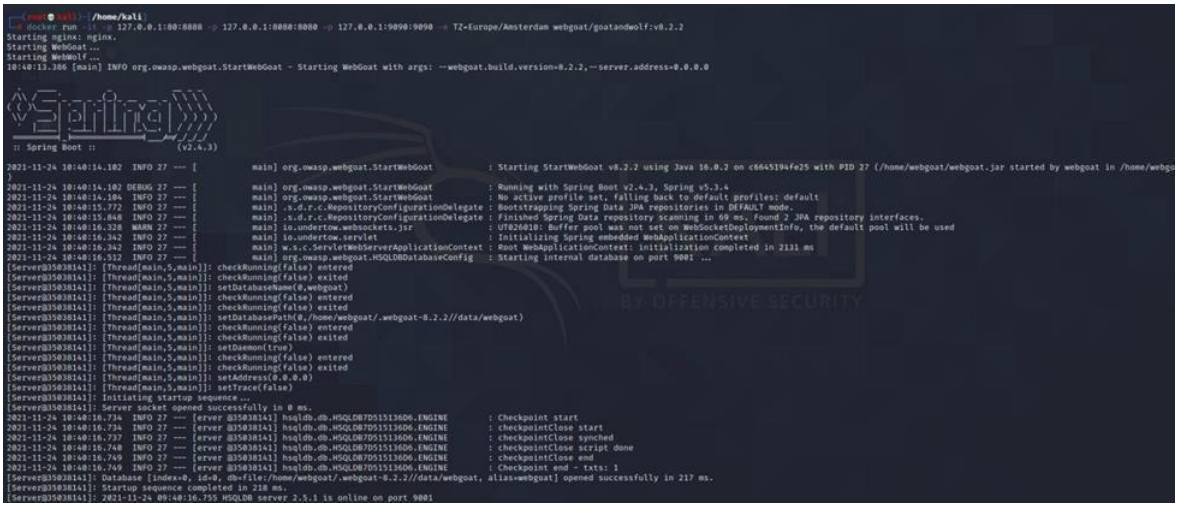


Рис. 3.5 – Запуск зображення Docker WebGoat

На наступних рисунках можна побачити веб-інтерфейси для застосунків WebGoat (Рис. 3.6) та WebWolf (Рис. 3.7).

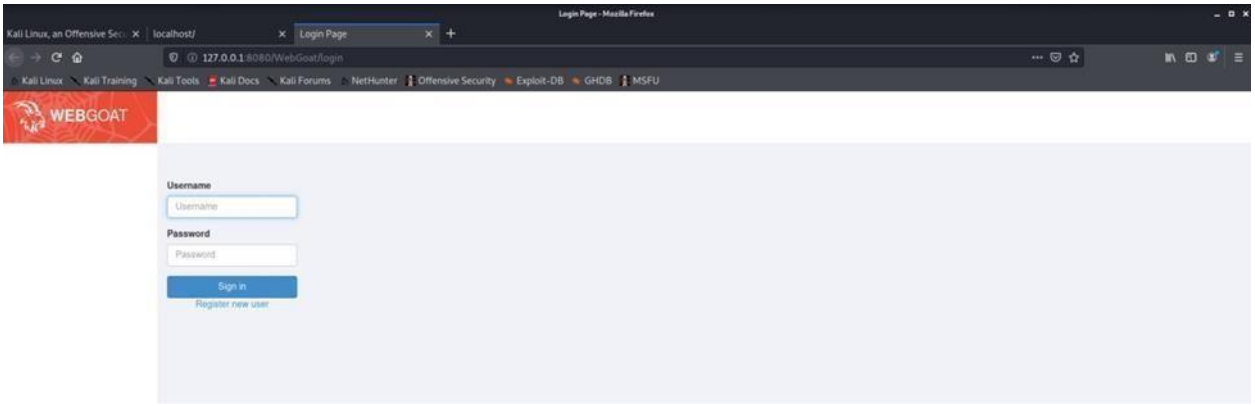


Рис. 3.6 – Інтерфейс застосунку WebGoat

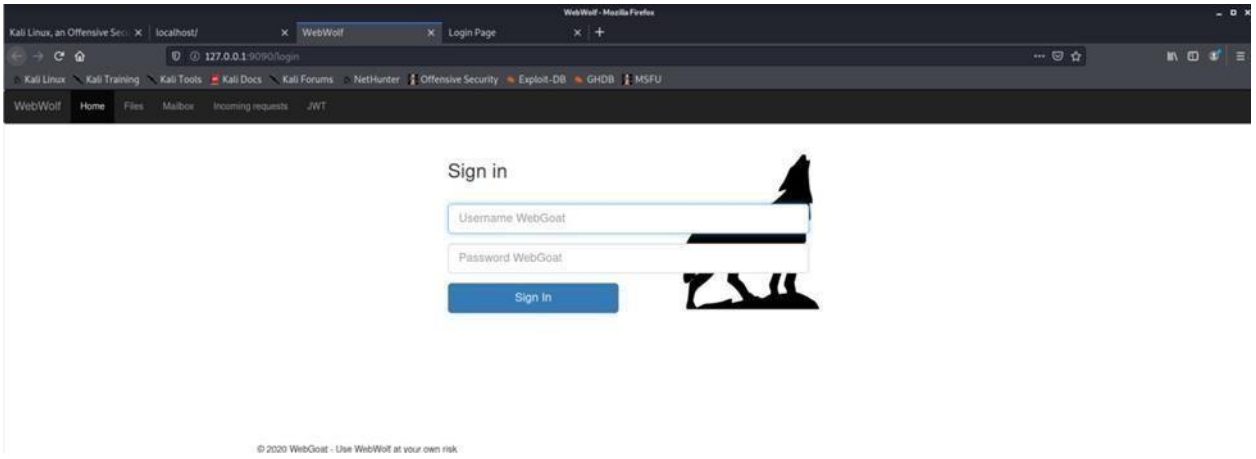


Рис. 3.7 – Інтерфейс застосунку WebWolf

Для проведення управління ризиками нам потрібно:

1. Ідентифікувати ризики.
2. Аналіз та оцінка ризиків за FMEA.
3. Складання плану тестування.
4. Виконання тесту.
5. Огляд результатів тесту.
6. Переоцінка ризиків.
7. План реагування на ризики.
8. Моніторинг.

1. Ідентифікація ризиків. Один з основних етапів ризик-орієнтованого тестування - це визначення потенційних ризиків, які можуть впливати на якість програмного продукту. Це може бути пов'язано з функціональністю, надійністю, безпекою, продуктивністю тощо.
2. Планування тестів. На основі ідентифікованих ризиків розробляється план тестування, який зорієнтований на виявлення та вирішення цих ризиків. Визначаються тести, які дозволяють перевірити функціонал або аспекти системи, які є найбільш ризиковими.
3. Виконання тестів. Тестові сценарії, які охоплюють ризиковані аспекти продукту, виконуються для виявлення потенційних проблем. Тестери можуть використовувати автоматизовані засоби або ручне тестування, залежно від природи тестів.

4. Оцінка результатів. Отримані результати тестування оцінюються щодо їхнього впливу на ризики. З'ясовується, чи були виявлені потенційні проблеми, чи вони були вирішені, і як це впливає на загальну якість продукту.
5. Звітність. Формується звіт, в якому фіксуються виявлені ризики, результати тестів та рекомендації щодо вирішення проблем. Цей звіт може служити основою для прийняття рішень щодо подальших етапів розробки або випуску продукту.
6. Моніторинг. Ризик-орієнтоване тестування може вимагати постійного моніторингу на протязі життєвого циклу продукту. Нові ризики можуть виникати або існуючі можуть змінюватися, і тестування повинно адаптуватися до цих змін.

На рисунку 3.8 можна побачити оцінку ідентифікованих ризики на етапі Збірки.

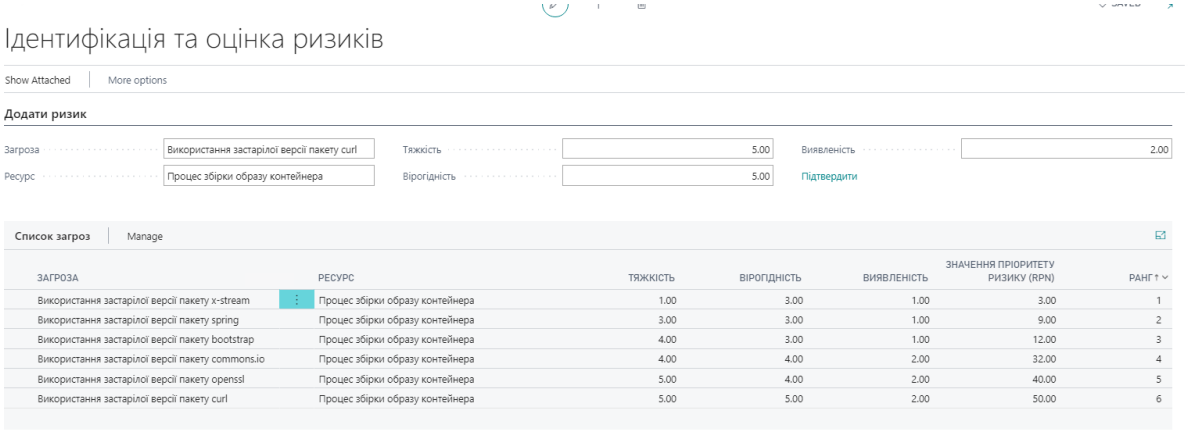


Рис. 3.8 – Оцінка ідентифікованих ризиків на етапі Збірки

Оскільки до нашого переліку загроз належать використання застарілих версій пакетів, то краще за все використати стандартну перевірку пакетів за допомогою сканеру Anchore Engine

Anchore Engine — це проект з відкритим кодом, який надає централізовану послугу з перевірки, аналізу та сертифікації зображень контейнерів. Anchore Engine надається як образ контейнера Docker, який можна запускати окремо або

в рамках платформи оркестровки, наприклад Kubernetes, Docker Swarm, Rancher, Amazon ECS та інших платформ для оркестровки контейнерів.

Крім того, також є кілька модульних інструментів-контейнерів, які можна запускати окремо або інтегрувати в автоматизовані робочі процеси, такі як конвеєри CI/CD.

Anchore постійно відстежує ваші репозиторії на наявність оновлень і повторює сканування за потреби.

Після визначення типу тесту та інструменту для його виконання ми можемо приступати до виконання тестів.

Для початку, підтягнемо з репозиторію образ цільової програми на Рис.3.9.

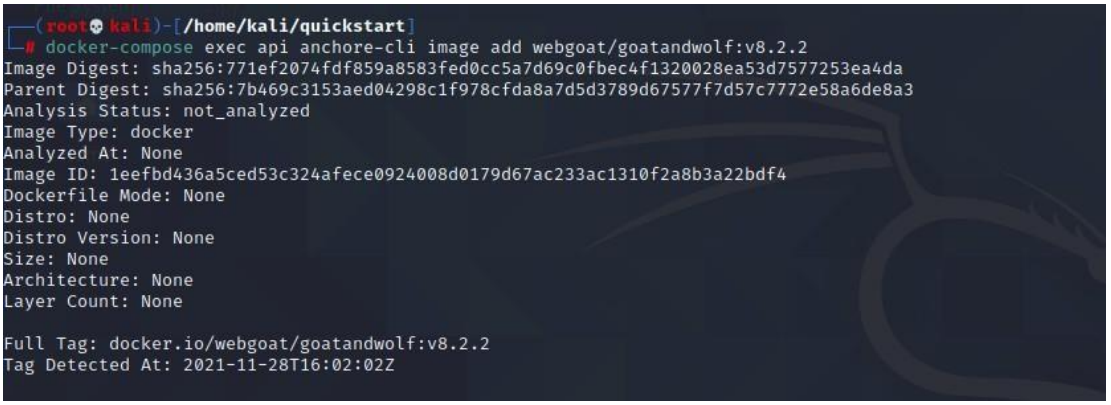


Рис. 3.9 – Підтягнення образу WebGoat до сканеру Anchore

Далі Анкор автоматично почне стандартний тест, і почавши 2 хвилини і 34 секунди ми отримуємо повідомлення, що образ просканировано (Рис. 3.10)

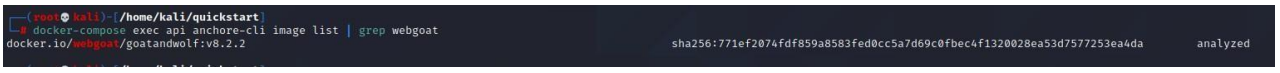
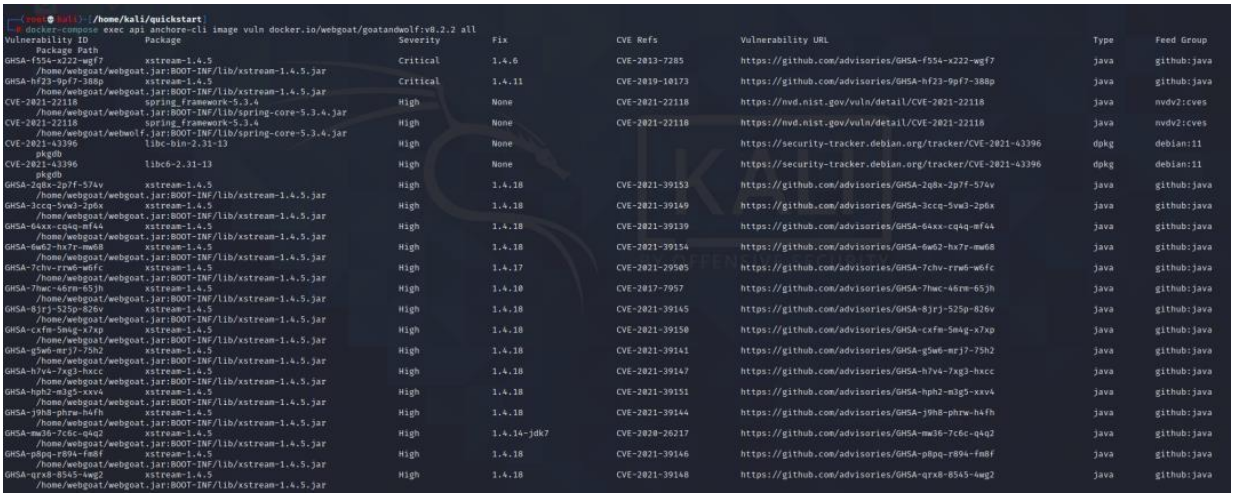


Рис. 3.10 – Статуси сканування образів

Після того, як образ протестовано ми можемо подивитись на результати (Рис. 3.11)



```
┌──(root㉿kali)-[/home/kali/quickstart]
└─# docker-compose exec api anchore-cli image vuln docker.io/webgoat/goatandwolf:v8.2.2 all
Vulnerability ID        Package                                                 Severity        Fix             CVE Refs        Vulnerability URL                                                       Type    Feed Group      Package Path
GHSA-f554-x222-wgf7     xstream-1.4.5                                           Critical        1.4.6           CVE-2013-7285   https://github.com/advisories/GHSA-f554-x222-wgf7                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-hf23-9pf7-388p     xstream-1.4.5                                           Critical        1.4.11          CVE-2019-10173  https://github.com/advisories/GHSA-hf23-9pf7-388p                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
CVE-2021-22118          spring_framework-5.3.4                                  High            None            CVE-2021-22118  https://nvd.nist.gov/vuln/detail/CVE-2021-22118                         java    nvdv2:cves      /home/webgoat/webgoat.jar:BOOT-INF/lib/spring-core-5.3.4.jar
CVE-2021-22118          spring_framework-5.3.4                                  High            None            CVE-2021-22118  https://nvd.nist.gov/vuln/detail/CVE-2021-22118                         java    nvdv2:cves      /home/webgoat/webwolf.jar:BOOT-INF/lib/spring-core-5.3.4.jar
CVE-2021-43396          libc-bin-2.31-13                                        High            None                            https://security-tracker.debian.org/tracker/CVE-2021-43396              dpkg    debian:11       pkgdb
CVE-2021-43396          libc6-2.31-13                                           High            None                            https://security-tracker.debian.org/tracker/CVE-2021-43396              dpkg    debian:11       pkgdb
GHSA-2q8x-2p7f-574v     xstream-1.4.5                                           High            1.4.18          CVE-2021-39153  https://github.com/advisories/GHSA-2q8x-2p7f-574v                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-3ccq-5vw3-2p6x     xstream-1.4.5                                           High            1.4.18          CVE-2021-39149  https://github.com/advisories/GHSA-3ccq-5vw3-2p6x                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-64xx-cq4q-mf44     xstream-1.4.5                                           High            1.4.18          CVE-2021-39139  https://github.com/advisories/GHSA-64xx-cq4q-mf44                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-6w62-hx7r-mw68     xstream-1.4.5                                           High            1.4.18          CVE-2021-39154  https://github.com/advisories/GHSA-6w62-hx7r-mw68                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-7chv-rrw6-w6fc     xstream-1.4.5                                           High            1.4.17          CVE-2021-29505  https://github.com/advisories/GHSA-7chv-rrw6-w6fc                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-7hwc-46rm-65jh     xstream-1.4.5                                           High            1.4.10          CVE-2017-7957   https://github.com/advisories/GHSA-7hwc-46rm-65jh                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-8jrj-525p-826v     xstream-1.4.5                                           High            1.4.18          CVE-2021-39145  https://github.com/advisories/GHSA-8jrj-525p-826v                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-cxfm-5m4g-x7xp     xstream-1.4.5                                           High            1.4.18          CVE-2021-39150  https://github.com/advisories/GHSA-cxfm-5m4g-x7xp                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-g5w6-mrj7-75h2     xstream-1.4.5                                           High            1.4.18          CVE-2021-39141  https://github.com/advisories/GHSA-g5w6-mrj7-75h2                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-h7v4-7xg3-hxcc     xstream-1.4.5                                           High            1.4.18          CVE-2021-39147  https://github.com/advisories/GHSA-h7v4-7xg3-hxcc                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-hph2-m3g5-xxv4     xstream-1.4.5                                           High            1.4.18          CVE-2021-39151  https://github.com/advisories/GHSA-hph2-m3g5-xxv4                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-j9h8-phrw-h4fh     xstream-1.4.5                                           High            1.4.18          CVE-2021-39144  https://github.com/advisories/GHSA-j9h8-phrw-h4fh                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-mw36-7c6c-q4q2     xstream-1.4.5                                           High            1.4.14-jdk7     CVE-2020-26217  https://github.com/advisories/GHSA-mw36-7c6c-q4q2                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-p8pq-r894-fm8f     xstream-1.4.5                                           High            1.4.18          CVE-2021-39146  https://github.com/advisories/GHSA-p8pq-r894-fm8f                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
GHSA-qrx8-8545-4wg2     xstream-1.4.5                                           High            1.4.18          CVE-2021-39148  https://github.com/advisories/GHSA-qrx8-8545-4wg2                       java    github:java     /home/webgoat/webgoat.jar:BOOT-INF/lib/xstream-1.4.5.jar
```

Рис. 3.11 – Вивід результатів Anchore

Як бачимо із виводу результатів сканера образу контейнера ми отримуємо список вразливостей, вразливий пакет, Тяжкість, чи є рішення проблеми та URL. Завдяки цій інформації процес аналізу результатів є надзвичайно простим і зрозумілим та займає приблизно 1 хвилину. Крім описаних вище вразливих пакетів сканер нам показав наявність ще декількох – libgnuts, ant, libgssapi. Також слід зазначити, що пакети curl та openssl не представлені в результатах скану, тобто є потрібної версії.

Наступним кроком ми додаємо знайдені ризики та переоцінюємо існуючі ризики та складаємо план реагування на ризики.

В даному випадку, для кожного ризику буде однакове рішення для реагування на ризик – оновлення пакетів до найновішої доступної версії кожного пакету.

Наступним кроком є моніторинг виявлених ризиків. Оскільки Анкор може відслідковувати зміни в репозиторіях на наявність оновлень і повторювати сканування за потреби, нам залишається дочекатись оновлень та порівняти результати двох сканувань.

Оскільки Анкор не має необхідної функціональності для порівняння результатів сканувань, було запропоновано створення скрипта, що допоможе порівняти старі результати скана та нові. Результатом використання даного скріпта стане вивід невиправлених CVE та їх кількості, виправлених CVE та їх кількості, а також нових знайдених CVE та їх кількості. (Рис. 3.12)

Рис. 3.12 – Вигляд виводу результатів виконання скрипта

Даний скрипт значно полегшує роботу та допомагає автоматизувати і спростити роботу при управлінні ризиків.

Сканування spider виявило 13 ресурсів (шлях http://webgoat:8080/WebGoat/) і активне сканування виявило 11 уразливостей. ZAP класифікував вразливості за ризиком, що призводить до 3 «Інформаційних»,4 «Низьких», 2 «Середніх» і 3 «Високих» рівнів ризику. Також було виявлено уразливість з високим ризиком і позначає цей ресурс як уразливий для SQLi. Результати зображені на рис. 3.13.

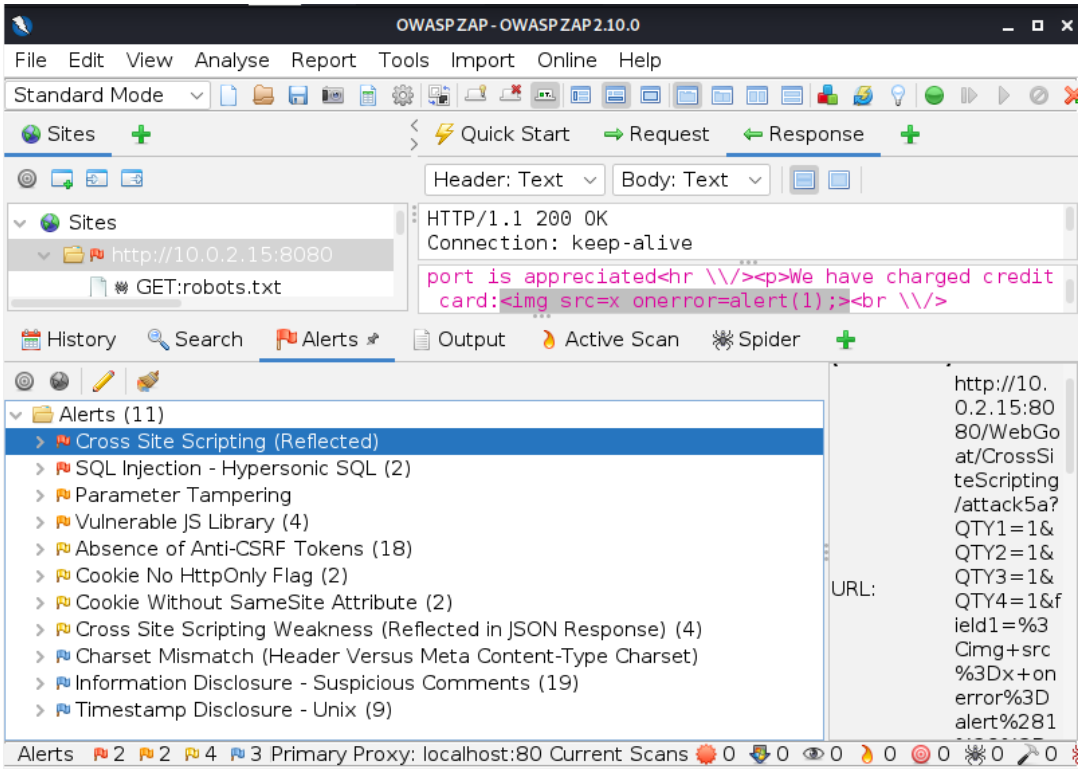


Рис. 3.13 – Результати WAST з використанням Spider та активного сканування

Тест SAS проводився проти http://webgoat:8080/WebGoat/login і ZAP виявив одну вразливу адресу. Вона піддається п’ятьом «низьким» рівням ризику і одному «середньому» рівню ризику. Результати зображені на рис. 3.14.

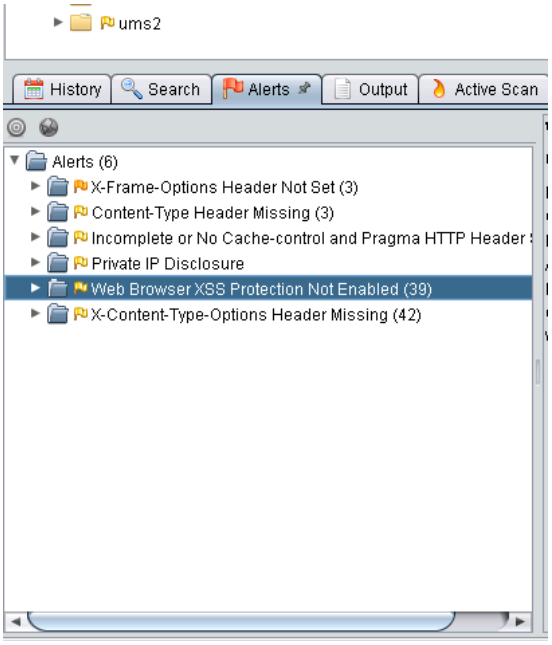


Рис. 3.14 – Результати SAS тестування

Під час проведених тестів, керованих поведінкою, ZAP виявив 10 уразливостей, які складаються з 3 «Інформаційних» і 4 «Низьких», 1 «Середнього» і 2 «Високих» рівнів ризику. Результати зображені на рис. 3.15.

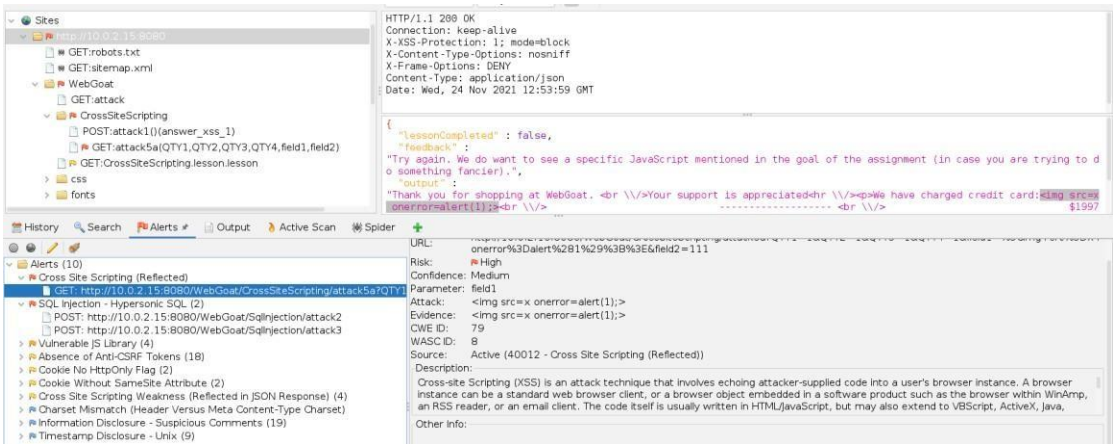


Рис. 3.15 – Результати BDST тестування

Ефективність ризик-орієнтованого тестування залежить від якості ідентифікації ризиків і адекватності тестових сценаріїв для їх виявлення.

Висновки до третього розділу

Розглядалося створення програмного комплексу для оцінки ризиків за допомогою методу FMEA, а також були представлені результати його функціонування.

Ризик-орієнтоване тестування - це процес ідентифікації, аналізу та оцінки ризиків, пов'язаних із розробкою та впровадженням програмного забезпечення, та визначення тестових сценаріїв на основі цих ризиків.

Для перевірки вірності та ефективності функціонування системи управління ризиками та розробленої методики ризик-орієнтованого тестування в рамках DevSecOps, був проведений експеримент. Цей експеримент включав в себе впровадження управління ризиками та ризик-орієнтованого тестування на конвеєрі CI/CD з методології DevOps, зокрема на етапах Збірки та Тестування.

Застосовуючи розроблену методику ризик-орієнтованого тестування, був проілюстрований процес тестування та оцінки ризиків на прикладі простого застосунку. Використовуючи цей підхід, вдалося виявити, проаналізувати та оцінити потенційні кібербезпекові ризики, які існують у даному застосунку. На основі отриманих даних були розроблені плани тестування, які включали в себе як статичний, так і динамічний аналіз.

РОЗДІЛ 4
ЕКОНОМІЧНА ЧАСТИНА

4.1 Організаційно-економічне та маркетингове обґрунтування проекту

Метою цього розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності від розробки, а також прийняття рішення щодо подальшого розвитку і впровадження або ж недоцільність впровадження відповідної розробки.

Для здійснення оцінки потрібно зробити розрахунки трудомісткості кожної операції, що мала місце при проведенні наукових досліджень.

Витрати часу по окремих операціях технологічного процесу відображені в табл. 4.1.

Таблиця 4.1

Операції технологічного процесу та час їх виконання

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1.	Витрати праці на підготовку опису задачі	інженер	11
2.	Витрати праці на розробку проекту	інженер	45
3.	Витрати праці на розробку структури системи	інженер	25
4.	Витрати праці на створення системи по вибраному проекту та структурі	інженер	70
5.	Витрати праці на підготовку документації	інженер	20
6.	Витрати праці на відлагодження роботи спроектованої системи при комплексній відладці	інженер	30
Разом			201

Сумарний час на проведення науково-дослідної роботи становить 201 годину.

Відповідно до Закону України “Про оплату праці” заробітна плата – це “винагорода, обчислена, як правило, у грошовому виразі, яку власник або уповноважений ним орган виплачує працівникові за виконану ним роботу”.

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства. Заробітна плата складається з основної та додаткової оплати праці.

Основна заробітна плата нараховується на виконану роботу за тарифними ставками, відрядними розцінками чи посадовими окладами і не залежить від результатів господарської діяльності підприємства.

Додаткова заробітна плата – це складова заробітної плати працівників, до якої включають витрати на оплату праці, не пов'язані з виплатами за фактично відпрацьований час. Нараховують додаткову заробітну плату залежно від досягнутих і запланованих показників, умов виробництва, кваліфікації виконавців. Джерелом додаткової оплати праці є фонд матеріального стимулювання, який створюється за рахунок прибутку.

При розрахунку заробітної плати кількість робочих днів у місяці слід в середньому приймати – 24,5 дні/міс., або ж 196 год./міс. (тривалість робочого дня – 8 год.).

Рекомендовані тарифні ставки: керівник проекту – 30,00...50,00 грн./год., інженер – 19,34...30,00 грн./год., консультант – 19,34...30,00 грн./год., технік – 19,34...30,00 грн./год., лаборант – 19,34...25,00 грн./год.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с \cdot К_г, \quad (4.1)$$

де $Т_с$ – тарифна ставка, грн.; $К_г$ – кількість відпрацьованих годин.

Оскільки всі види робіт в даному дослідженні виконує інженер, то основна заробітна плата буде розраховуватись тільки за однією формулою

$$З_{осн} = 30 \cdot 201 = 6030 \text{ грн.}$$

Додаткова заробітна плата становить 10–15 % від суми основної заробітної плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.}, \quad (4.2)$$

де $К_{допл.}$ – коефіцієнт додаткових виплат працівникам, 0,1–0,15 (візьмемо його рівним 0,13).

$$З_{дод.} = 6030 \cdot 0,13 = 783,9$$

грн.

Звідси загальні витрати на оплату праці ($В_{о.п.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.}. \quad (4.3)$$

$В_{о.п}$=6030 +783,9 = 6813,9 грн.

Крім того, слід визначити відрахування на соціальні заходи:

- єдини соціальний внесок ЄСВ (прибутковий податок) - 22%;
- військовий збір – 1,5%.

У сумі зазначені відрахування становлять 23,5 %

Отже, сума відрахувань на соціальні заходи буде становити:

$$В_{с.з.} = ФОП \cdot 0,235, \quad (4.4)$$

де $ФОП$ – фонд оплати праці, грн.

$$В_{с.з.} = 6813,9 \cdot 0,235 = 1601,26 \text{ грн.}$$

Проведені розрахунки витрат на оплату праці зведемо у табл. 4.2.

Таблиця 4.2

Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, тис.грн.	Нарахув. на ФОП, тис.грн.	Всього витрати на оплату праці, тис.грн. *6=3+4+5*
		Тарифна ставка, тис.грн.	К-сть відпрацьов. год.	Фактично нарах. з/пл., тис.грн.			
А	*Б*	*1*	*2*	*3*	*4*	*5*	*6*
1	інженер	30	201	6030	783,9	1601,26	8415,16

Сумарні затрати на оплату праці становлять 8415,16 тис.грн.

Матеріальні витрати визначаються як добуток кількості витрачених матеріалів та їх ціни:

$$М_{Вi} = q_i \cdot p_i, \quad (4.5)$$

де q_i – кількість витраченого матеріалу *i*-го виду; p_i – ціна матеріалу *i*-го виду.

Звідси, загальні матеріальні витрати можна визначити:

$$З_{м.в.} = \sum M_{Bi}. \quad (4.6)$$

Проведені розрахунки занесемо у табл. 4.3.

Таблиця 4.3

Зведені розрахунки матеріальних витрат

Найменування матеріальних ресурсів	Одиниця виміру	Норма витрат	Ціна за одиницю, тис.грн	Затрати матеріалів, тис.грн	Транс- портно- заготівельні витрати, тис.грн	Загальна сума витрат на матеріали, тис.грн
1. Основні матеріали						
Програмне забезпечення різного рівня	комплект	1	4299	–	–	4299
Разом:						4299

Сумарні матеріальні затрати становлять 4299 тис.грн.

Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$З_e = W \cdot T \cdot S, \quad (4.7)$$

де W – необхідна потужність, кВт;

T – кількість годин роботи обладнання;

S – вартість кіловат-години електроенергії.

Вартість кіловат-години електроенергії слід приймати згідно існуючих на даний час тарифів (1,69 грн. з ПДВ за 1 кВт).

Потужність комп'ютера для проведення дослідження – 550 Вт, кількість годин роботи обладнання згідно таблиці 5.1 – 201 годин.

Тоді, $З_e$= 0,55 · 201 · 1,69 = 186,82 тис.грн

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення.

Комп'ютери та оргтехніка належать до четвертої групи основних фондів. Для цієї групи річна норма амортизації дорівнює 60 % (квартальна – 15 %).

Для визначення амортизаційних відрахувань застосовуємо формулу:

$$A = \frac{Б_В \cdot Н_А}{100\%}, \quad (4.8)$$

де A – амортизаційні відрахування за звітний період, тис.грн.; $Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, тис.грн.; $Н_А$ – норма амортизації, %.

Для даного дослідження засобом роботи є комп'ютер. Його сума становить 10650 тис.грн. Отже, амортизаційні відрахування будуть рівні:

$$\text{А} = \frac{10650 \cdot 5\%}{100\%} = 532{,}5 \text{ тис. грн.}$$

Оскільки робота виконувалась 201 годин, то амортизаційні відрахування будуть становити:

$$\text{А} = \frac{532{,}5 \cdot 201}{201} = 532{,}5$$

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління спілкою та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$Н_в = В_{о.п.} \cdot 0{,}2...0{,}6, \quad (4.9)$$

де $Н_В$ – накладні витрати.

Отже, накладні витрати:

$$Н_\text{В} = 6813{,}9 \cdot 0{,}5 = 3406{,}95 \text{ тис. грн.}$$

4.2 Розрахунки економічної ефективності впровадження нового програмного продукту

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Економічна ефективність (E_p) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_p = \frac{П}{C_B}, \tag{4.10}$$

де $П$ – прибуток; C_B – собівартість.

Плановий прибуток ($П_{пл}$) знаходимо за формулою:

$$П_{пл} = Ц - C_B. \tag{4.11}$$

Розраховуємо плановий прибуток:

$$\text{П}_{\text{пл}} = 26271{,}22 - 16840{,}53 = 9430{,}69 \text{ тис.грн}$$

Отже, формула для визначення економічної ефективності набуде вигляду:

$$E_p = \frac{П_{пл}}{Cв}. \tag{4.12}$$

Тоді, $\text{E}_{\text{p}} = \frac{9430{,}69}{16840{,}53} = 0{,}56$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень (T_p):

$$T_p = \frac{1}{E_p}, \tag{4.13}$$

Термін окупності дорівнює:

$$\text{T}_{\text{p}} = \frac{1}{0{,}56} = 1{,}79 \text{ року}$$

Розраховане значення економічної ефективності становить 0,56, що є високим значенням.

Для даного дослідження термін окупності становить 1,79 року.

Таблиця 4.4

Техніко-економічні показники НДР

№ п/п	Показник	Значення
1.	Собівартість, тис.грн.	16840,53
2.	Плановий прибуток, тис.грн.	9430,69
3.	Ціна, тис.грн.	26271,22
4.	Економічна ефективність	0,56
5.	Термін окупності, рік	1,79

Отже, дане дослідження може бути впроваджене та мати подальший розвиток, оскільки воно є економічно вигідним за всіма основними техніко- економічними показниками.

4.3 Визначення капітальних витрат

Результати проведених вище розрахунків зведемо у табл. 4.5.

Таблиця 4.5

Кошторис витрат на НДР

Зміст витрат	Сума, тис.грн.	В % до загальної суми
Витрати на оплату праці (основну і додаткову заробітну плату)	6813,9	40,46
Відрахування на соціальні заходи	1601,26	9,50
Матеріальні витрати	4299	25,52
Витрати на електроенергію	186,92	1,10
Амортизаційні відрахування	532,5	3,16
Накладні витрати	3406,95	20,23
Собівартість	16840,53	100

Собівартість (C_B) НДР розрахуємо за формулою:

$$C_B = B_{о.п.} + B_{с.з.} + З_{м.в.} + З_е + А + Н_в \cdot \quad (4.14)$$

Отже, собівартість дослідження дорівнює:

$$С_В = 6813{,}9 + 1601{,}26 + 4299 + 186{,}92 + 532{,}5 + 3406{,}95$$
$$= 16840{,}53 \text{ тис. грн.}$$

В результаті проведених розрахунків собівартість науково-дослідної роботи становить 16840,53 тис.грн.

Ціну НДР можна визначити за формулою:

$$Ц = \frac{C_в \cdot \left(1 + P_{рен}\right) + K \cdot B_{н.і.}}{K} \cdot \left(1 + ПДВ\right), \quad (4.15)$$

де $P_{рен.}$ – рівень рентабельності, 30 %; K – кількість замовлень, од. (встановлюється лише при розробці програмного продукту та мікропроцесорних систем); $B_{н.і.}$ – вартість носія інформації, тис.грн. (встановлюється лише при розробці програмного продукту); *ПДВ* – ставка податку на додану вартість, (20 %).

Оскільки розробка є прикладною, і використовуватиметься тільки для одного підприємства, то для розрахунку ціни не потрібно вказувати коефіцієнти K та $B_{і.н}$, оскільки їх в даному випадку не потрібно.

Тоді, формула для обчислення ціни НДР буде мати вигляд:

$$Ц = C_B \cdot \left(1 + P_{рен}\right) \cdot \left(1 + ПДВ\right). \quad (4.16)$$

Звідси ціна на НДР складе:

$$Ц = 16840{,}53 \cdot (1 + 0{,}3) \cdot (1 + 0{,}2) = 26271{,}22 \text{ тис. грн.}$$

Висновки до четвертого розділу

Метою цього розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності від розробки, а також прийняття рішення щодо подальшого розвитку і впровадження або ж недоцільність впровадження відповідної розробки.

Сумарний час на проведення науково-дослідної роботи становить 201 годину.

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та господарської діяльності підприємства. Заробітна плата складається з основної та додаткової оплати праці. Сумарні матеріальні затрати становлять 4299 тис.грн.

Дане дослідження може бути впроваджене та мати подальший розвиток, оскільки воно є економічно вигідним за всіма основними техніко-економічними показниками.

РОЗДІЛ 5 ОХОРОНА ПРАЦІ

5.1 Аналіз умов праці на робочому місці програміста

У приміщенні, де знаходиться робоче місце програміста, всього знаходяться десять робочих місць. Зазвичай одночасно працюють 10 працівників. Усі розрахунки будуть проводитися з урахуванням максимальної кількості робочих місць. Модель ПК встановлені в офісі використовують спеціальні монітори LCD, ПК має такі характеристики:

- процесор – Intel Pentium 3550M (2.3 ГГц);
- об'єм оперативної пам'яті – 4 ГБ;
- об’єм HDD – 500 ГБ;
- оптичний привід – DVD+/–RW.

Загальна площа приміщення становить $S = 8м * 8м = 64м^2$, об’єм $V = 8м * 8м * 3,5м = 224м^3$. Отже, в даному приміщенні площа на одне робоче місце становить приблизно $6,4м^2$, а об’єм – $22,4м^3$.

Площа на одне робоче місце повинна становити не менше ніж $6м^2$, об’єм – не менше $20м^3$, в приміщенні, що розглядається ці показники відповідають нормі ДСан ПиН 3.3.2-007-98.

Приміщення обладнане трифазною чотирьохпровідною мережею електроживлення з глухозаземленою нейтраллю. Напруга в електромережі 380/220 В, частота – 50 Гц.

Схематичне представлення системи «Л-М-С» дозволяє проаналізувати вплив шкідливих факторів на організм людини. Елементи системи можно розділити на такі функціональні частини:

«Людина» - працівник виробництва:

Л1 - людина, яка виконує управління “машиною”;

Л2 - людина, з точки зору безпосереднього впливу на навколишнє середовище;

Л3 - людина, яка розглядається з точки зору її фізіологічного стану.

«Машина» - обладнання в приміщенні:

М1 - елемент, який виконує основну технологічну функцію;

М2 – машина, що виконує функції аварійного захисту (занулення, ізоляція);

М3 - машина, що впливає на стан виробничого середовища та людину (вплив на людину, шумового та електромагнітного забруднення середовища персональним комп'ютером).

Наведемо структурну схему для розглянутого приміщення, рисунок 5.1.

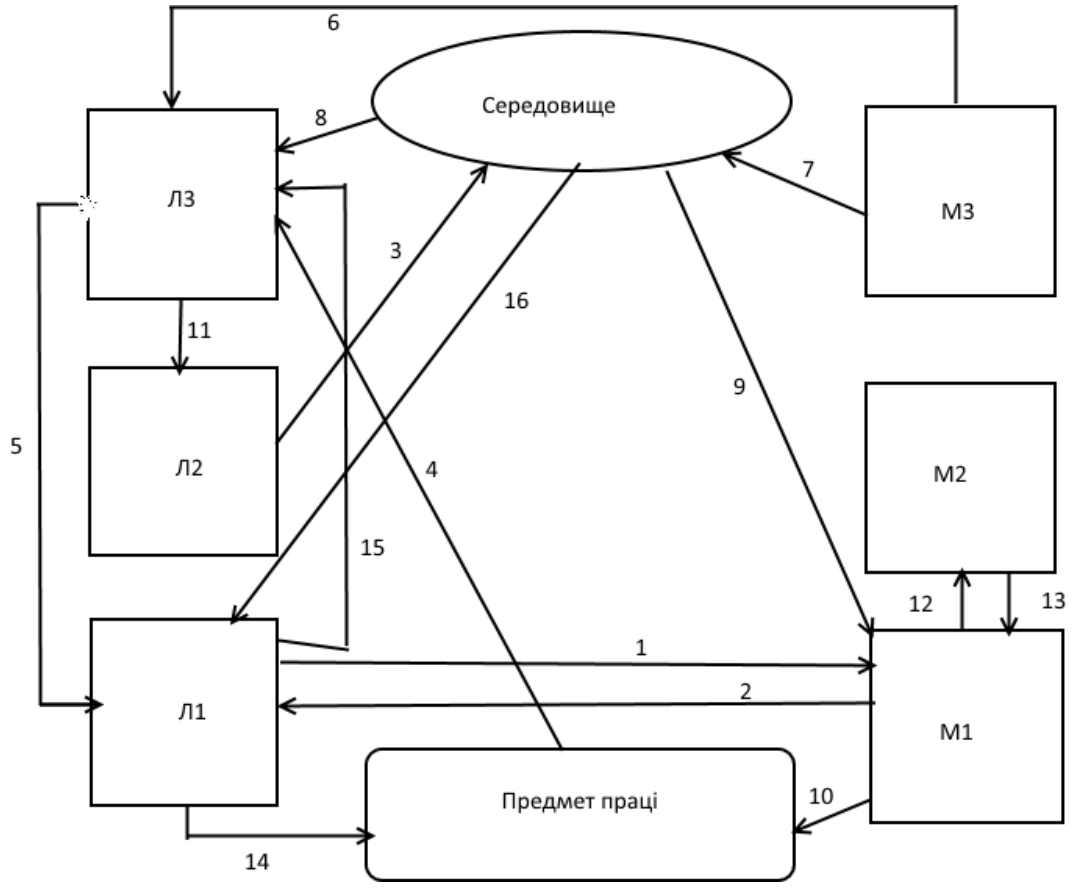


Рис. 5.1 – Схема Л-М-С для лабораторії

Отже, ми провели декомпозицію системи Л-М-С на сім елементів та виділили зв'язки між ними.

Таблиця 5.1

Перелік зв’язків у загальній системі Л-М-С

Номер зв'язку	Напрямки зв'язку	Зміст зв'язку
1	Л1-М1	вплив людини на персональний комп’ютер з метою створення предмета праці, програмного забезпечення
2	М1-Л1	Інформація про стан машини, про об'єкт праці й середовище
3	Л2-С	Вплив людини як біологічного об'єкта на середовище (виділення вуглекислого газу, тепловиділення)
4	ПП-Л3	Вплив предмета праці на психофізіологічний стан людини (розумова перенапруга, перенапруга аналізаторів)
5	Л3-Л1	Вплив стану організму людини на якість його роботи(емоційні перевантаження, перенапруження аналізаторів)
6	М3-Л3	Людина під впливом небезпечних виробничих чинників
7	М3-С	Цілеспрямований вплив машини на середовище (підвищена яскравість світла, підвищена температура, електромагнітне випромінювання)
8	С-Л3	Вплив виробничого середовища на стан організму (підвищена або знижена температура повітря на робочому місці, недостатнє висвітлення, підвищений шум)
9	С-М1	Вплив середовища на роботу машин (вологість і температура повітря впливають на працездатність машин)
10	М1-ПП	Вплив машини на предмет праці (підвищена напруженість електричного поля)
11	Л3-Л2	Вплив фізіологічного стану на ступінь інтенсивності обміну речовин між організмом і середовищем, і енерговиділенням людини (фізичні перевантаження, нервово психічні перевантаження
12	М1-М2	Інформація, необхідна для створення аварійних керуючих впливів (підвищене значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини)
13	М2-М1	Аварійні управляючі впливи (підвищене значення напруги в електричному ланцюзі, замикання якого може відбутися через тіло людини)

14	Л1-ПП	Вплив людини на ПП (налагоджування програми
15	Л1-Л3	Вплив виконуючих дій на фізіологічний стан людини
16	С-Л1	Вплив навколишнього середовища на якість роботи розробника (підвищена або знижена температура повітря на робочому місці, підвищена або знижена вологість повітря робочої зони, підвищена або знижена рухливість повітря в робочій зоні)
А	Зовнішня система управління –Л1	Керуюча інформація про технологічний процес з зовнішньої системи управління

Згідно ГОСТ 12.0.003-74 на робочому місці програміста присутні такі потенційно небезпечні та шкідливі виробничі чинники:

1.Фізичні:

- підвищена або знижена температура поверхонь обладнання (ПК);
- підвищена або знижена температура повітря робочої зони;
- підвищений рівень шуму на робочому місці;
- підвищена або знижена вологість повітря;
- підвищена або знижена рухливість повітря;
- підвищений рівень статичної електрики;
- підвищений рівень електромагнітного випромінювання
- відсутність або нестача природного світла;
- недостатня освітленість робочої зони;
- підвищена яскравість світла;
- знижена контрастність.

– підвищений рівень статичної електрики;

– підвищений рівень електромагнітного випромінювання

– відсутність або нестача природного світла;

– недостатня освітленість робочої зони;

– підвищена яскравість світла;

– знижена контрастність.

2. Хімічні чинники відсутні.
3. Біологічні чинники відсутні.
4. Психофізіологічні:

– нервово-психічні перевантаження.

– монотонність праці;

– розумове перенапруження;

– перенапруження зору.

Домінуючим шкідливим фактором є: підвищена температура повітря на робочому місці.

5.2 Промислова безпека у виробничому приміщенні

Характеристики ПК, встановлених на робочих місцях, повністю відповідають вимогам чинних в Україні стандартів. Все обладнання, встановлене на робочих місцях, а також у приміщенні вцілому, є справним.

За ступенем безпеки поразки електричним струмом, згідно ПУЕ - 2011, приміщення належить до класу приміщень без підвищеної небезпеки, тому що виключено можливість одночасного дотику людини до корпусів електроустаткування й заземлених металевих конструкцій приміщення.

Необхідно проводити контроль ізоляції. Контроль проводити між нулем і фазою і між фазами. Опір ізоляції має бути не менше 0,5 МОм. Контроль проводити не рідше 1 разу на рік при відключеному електроживленні.

Електроживлення ПК у приміщенні проводиться від розеток типу "Європа" з заземлюючими контактами. Всі електричні розетки, мають маркування по напрузі. Напруга в розетках такого типу складає $220В$, частота струму - $50Гц$.

Для захисту людей від ураження електричним струмом передбачена система заземлення типу TN, оскільки використовується електрична мережа напругою до 1000 В з глухо заземленою нейтраллю згідно НПАОП 0.00-1-28.

Захисний ефект заземлення типу TN полягає в зменшенні тривалості замикання на корпус, отже, у скороченні часу впливу електричного струму на людину. Це досягається з'єднанням металевих корпусів устаткування з нульовим захисним проводом мережі. Нульовий провід підлягає повторному заземленню.

Опір повторного заземлення нульового провідника повинен бути не більше 10 Ом, а перетин нульового провідника мережі має бути такий ж, як і перетин фазного проводу. Опір ізоляції, має бути не менше 0,5 МОм. Контроль ізоляції здійснюється 1 раз на рік відповідно до ПУЕ-2011. Також у приміщенні є аварійний резервний вимикач, що може повністю вимкнути електричне живлення приміщення, крім освітлення, який повинен бути встановлений у приміщенні, де одночасно експлуатуються понад п’ять ПК, відповідно до норм НПАОП 40.1-1-32-01.

5.3 Виробнича санітарія у приміщенні

Джерелом природного освітлення є вікна, що виходять на північ. Також в приміщенні використовується загальне штучне освітлення. Штучне висвітлення необхідно виконати у вигляді суцільних або переривчастих ліній світильників, розташованих паралельно лінії зору операторів. Освітленість при роботі з екраном у сполученні з роботою над документами повинна бути не менш 300 лк. В якості джерел світла рекомендується використовувати люмінесцентні лампи.

Зорова робота оператора ПЕВМ є роботою високої точності, оскільки найменший розмір об'єкта розрізнення 0.3 - 0.5 мм і розряд зорової роботи – III. За вимог ДБН В. 2.5-28-2006 величина коефіцієнта природної освітленості (КПО)

повинна бути більше 1,2%. Природне світло проникає в приміщення НДЛ через бічні світлові прорізи, що відповідає вимогам. Вікна повинні мати жалюзі або штори.

Температура внутрішніх поверхонь приміщень, таких як стіни, підлога, стеля, а також температура зовнішніх поверхонь технологічного устаткування не повинна виходити за межі допустимих величин температури повітря.

Згідно ГОСТ 12.1.005-88 і ДСН 3.3.6.042-99 робота відноситься до категорії 1а легкої і для неї встановлені відповідні нормативні параметри.

Відповідно нормам ДСН 3.3.6.042-99, оптимальні та допустимі показники мікроклімату у приміщенні для роботи програміста наведені у таблиці 5.2.

Таблиця 5.2

Оптимальні та допустимі показники мікроклімату для роботи програміста

	Період року	Температура повітря, град. С	Відносна вологість, %	Швидкість руху, м/с
Оптимальні показники	Теплий	23 − 25	60 – 40	0,1
	Холодний	22 − 24	60 – 40	0,1
Допустимі показники	Теплий	22 − 28	55 − при 28 град. С	0,2 − 0,1
	Холодний	21 − 25	75	не більше 0,1

На нашому робочому місці середня температура дорівнює приблизно 30 градусів. Температура не відповідає ДСН 3.3.6.042-99, тому ми маємо прийняти міри.

Нам треба знайти, чому на робочому місці підвищена температура повітря. Для цього ми розраховуємо виділення тепла при роботі обладнання.

$$Qобл = 860 * N * K1 * K2;$$

N – потужність одиниці обладнання, Вт (N = 150 Вт для комп'ютера, N = 50 Вт для принтера, ксерокса, сканера та ін.). k_1 – коефіцієнт використання встановленої потужності, k_1 =0,8; k_2 – коефіцієнт одночасної роботи обладнання, k_2 =0,5.

Розрахуємо потужність одиниці обладнання комп'ютера та принтера.

$$N = 10 * 150 + 50 = 1550$$

Виділення тепла при роботі обладнання.

$Qобл = 860 * 1{,}55 * 0{,}8 * 0{,}5 = 533{,}2$*(ккал/г)*

Ми можемо зробити висновок, що значення 533,2ккал/г завелике для тепловиділення від 10 персональних комп'ютерів. Потрібно захистити людей від підвищеної температури. Для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції.

5.4 Пожежна безпека виробничого приміщення

Як відомо пожежа може виникнути при взаємодії горючих речовин, окислення і джерел запалювання.

До горючих речовин у приміщенні, що розглядається, можна віднести: будівельні матеріали для акустичної і естетичної обробки приміщень, двері, підлога, столи, крісла, ізоляція кабелів, папір і т.п.

Джерелами запалювання можуть бути електронні схеми ПК та інших приладів, таких як принтери, сканери, ксерокси і т.п., пристрої електроживлення, кондиціонування повітря, де в результаті різних порушень утворюються перегріті елементи, електричні іскри і дуги, здатні викликати загоряння горючих матеріалів.

У сучасних ПК дуже висока щільність розміщення елементів електронних схем. У безпосередній близькості один від одного розташовуються проводи, кабелі. При протіканні по них електричного струму виділяється значна кількість теплоти, при цьому можливо оплавлення ізоляції. Для відводу надлишкової теплоти в ПК встановлені системи вентиляції та кондиціонування повітря, які при постійній роботі самі створюють додаткову пожежну небезпеку.

Горючі гази та легкозаймисті рідини у приміщенні відсутні. Проте, враховуючи наявність потенційних джерел запалювання, у вигляді ПК та інших

додаткових приладів, та горючих речовин, згідно НАПБ Б.03.002 - 2007, приміщення, в якому знаходяться ПК, можна віднести до категорії В (пожежонебезпечні). Будівля, в якій знаходиться приміщення має *II* ступінь вогнестійкості, згідно ДБН В.1.1.7-2002. Ступінь вогнестійкості будинку, де розташовується приміщення, відповідає пожежовибухонебезпеці виробництва.

Відповідно до вимог нормативно-правових актів з питань пожежної безпеки України (НАПБ А.01 001-2004); приміщення має бути оснащене:

– димовими пожежними сповіщувачів у кількості 1 штука, згідно ДБН В.2.5-56:2010;
– кошма;
– вуглекислотними переносними вогнегасниками типу ввк ємністю не менше 1,4 кг.

Лабораторне приміщення належить до категорії пожежної небезпеки В, так як в ньому відсутні горючі гази і рідини. Тому приміщення даного типу необхідно оснащувати вуглекислотними переносними вогнегасниками типу ВВК- 1, 4. Площа лабораторного приміщення дорівнює 91 кв.м. Виходячи з вимог НАПБ Б.03.001 - 2004 (1 вогнегасник на 20 кв.м площі, але не менше 2 вогнегасників у приміщенні) кількість необхідних вогнегасників в лабораторії повинна дорівнювати 4.

В будинку, в якому знаходиться приміщення, що розглядається, наявні евакуаційні виходи, що відповідають нормам ДБН В.1.1.7-2002. Найближчий евакуаційний вихід знаходиться в сусідньому приміщенні, шлях евакуації проходить через коридор та сусіднє приміщення. Максимальна відстань до евакуаційного виходу з найвіддаленішої точки підлоги в приміщенні, що розглядається, близько.

Ширина проходів до одиничних робочих місць у приміщенні не менше 1*м* Згідно ДБН В.1.1.7-2002, така ширина проходів всередині приміщення є допустимою.

Висновки до п’ятого розділу

У приміщенні, де знаходиться робоче місце програміста, всього знаходяться десять робочих місць. Зазвичай одночасно працюють 10 працівників.

Домінуючим шкідливим фактором є: підвищена температура повітря на робочому місці.

Характеристики ПК, встановлених на робочих місцях, повністю відповідають вимогам чинних в Україні стандартів. Все обладнання, встановлене на робочих місцях, а також у приміщенні вцілому, є справним.

Ми можемо зробити висновок, що значення 533,2ккал/г завелике для тепловиділення від 10 персональних комп’ютерів. Потрібно захистити людей від підвищеної температури. Для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції.

В будинку, в якому знаходиться приміщення, що розглядається, наявні евакуаційні виходи, що відповідають нормам ДБН В.1.1.7-2002. найближчий евакуаційний вихід знаходиться в сусідньому приміщенні, шлях евакуації проходить через коридор та сусіднє приміщення. Максимальна відстань до евакуаційного виходу з найвіддаленішої точки підлоги в приміщенні, що розглядається, близько.

ЗАГАЛЬНІ ВИСНОВКИ

Провівши дослідження, можемо зробити наступні висновки:

1. З інтенсивним розвитком технологій програмне забезпечення стає все більш складним, що призводить до виникнення нових методологій розробки. Для забезпечення якості на протязі всього циклу розробки важливо використовувати гнучкі методології розробки. У даному випадку були розглянуті та порівняні такі методології, як DevOps, Agile та Waterfall.

Кожна з цих методологій має свої переваги та особливості. DevOps вигідний завдяки автоматизації та швидкому розгортанню, Agile наголошує на гнучкості та співпраці, тоді як Waterfall пропонує послідовний та докладний підхід. Вибір методології залежить від конкретних потреб та характеристик проекту.

2. Ризик-орієнтований підхід допомагає ефективно використовувати ресурси, скорочує час тестування та підвищує ймовірність виявлення та усунення критичних проблем. Це особливо корисно в умовах розвитку проектів з великою динамікою та обмеженими ресурсами.

Щоб переконатись в коректності управління ризиками був проведений експеримент для виявлення кращого аналізу та оцінки ризиків. В ході дослідження були взяті наступні методики – DREAD Microsoft, PrisMA, PRAM, FMEA та FTA.

Важливим аспектом в методології DevOps є оформлення результатів, яке повинно підвищити розуміння видів ризиків, їх можливого впливу та шляхів обробки. Обрана методика повинна забезпечити простежуваність, відтворюваність та верифікацію процесу та результатів.

3. Під час проведення досліджень було встановлено, що методика оцінки ризиків DREAD виходить на лідерські позиції за всіма показниками, у той час як FMEA та FTA знаходяться на другому місці. Однак, враховуючи вимоги методології розробки DevOps до оперативного управління ризиками, виявлено,

що методика DREAD не є найкращим вибором через свою недостатню швидкість.

З цього приводу пропонується використовувати методику FMEA, яка продемонструвала кращі результати і, водночас, має менше атрибутів для оцінки, що призводить до заощадження часу. Таким чином, методика FMEA вважається оптимальною з обраної перспективи, враховуючи вказані критерії.

DevSecOps дозволяє розробникам і безпековим експертам працювати разом в реальному часі, спільно вирішуючи завдання безпеки та враховуючи їх в процесі розробки. Це сприяє покращенню безпеки програмного забезпечення та ефективності відповіді на можливі інциденти.

4. Розглядалося створення програмного комплексу для оцінки ризиків за допомогою методу FMEA, а також були представлені результати його функціонування. Ризик-орієнтоване тестування - це процес ідентифікації, аналізу та оцінки ризиків, пов'язаних із розробкою та впровадженням програмного забезпечення, та визначення тестових сценаріїв на основі цих ризиків.

Для перевірки вірності та ефективності функціонування системи управління ризиками та розробленої методики ризик-орієнтованого тестування в рамках DevSecOps, був проведений експеримент. Цей експеримент включав в себе впровадження управління ризиками та ризик-орієнтованого тестування на конвеєрі CI/CD з методології DevOps, зокрема на етапах Збірки та Тестування.

5. Метою економічного розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності від розробки, а також прийняття рішення щодо подальшого розвитку і впровадження або ж недоцільність впровадження відповідної розробки.

Сумарні матеріальні затрати становлять 4299 тис.грн.

Дане дослідження може бути впроваджене та мати подальший розвиток, оскільки воно є економічно вигідним за всіма основними техніко-економічними показниками.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бутузов В. М. Співвідношення понять «комп'ютерна злочинність» і «кіберзлочинність». Інформаційна безпека людини, суспільства, держави. 2014.
2. Войтович В.С., Гриник Р.О. Дослідження проблематики кібербезпеки України. Зб. наук. праць XII Міжнар. наук.-практ. конф. Молодих вчених, курсантів та студентів. Проблеми та перспективи розвитку системи безпеки життєдіяльності (м. Львів, 23-24 березня 2017 р.). [в 2 ч.]. Ч. 2. Львів: ЛДУ БЖД, 2017. С. 11–12.
3. Войтович В.С., Гриник Р.О. Основні безпекові проблеми кіберпростору України. Зб. тез доповідей Міжнародна науково-практична конференція Інформаційна безпека в сучасному суспільстві (м. Львів, 24-25 листопада 2016 р.). Львів : ЛДУБЖД, 2016. С. 23–24.
4. В чому різниця DevOps і DevSecOps. URL: https://dou.ua/lenta/articles/devops-security/ (дата звернення: 22.11.2023)
5. Гнатюк С. О. Кібертероризм: історія розвитку, сучасні тенденції та контрзаходи. *Безпека інформації*, 2013.
6. Додавання Sec в DevSecOps. URL: https://www.megatrade.ua/news/reviews/devsecops-ta-secops-shcho-tse-i-yak-vprovadzhuvati/ (дата звернення: 22.11.2023)
7. Історія створення DevOps. URL: https://dou.ua/forums/topic/41367/ (дата звернення: 22.11.2023)
8. IT – безпека. URL: https://astwellsoft.com/uk/blog/software-security.html (дата звернення: 22.11.2023)
9. Колісніченко О. А., Коломицев М. В. Модель ризик-орієнтованого тестування веб-застосунків. URL:https://ela.kpi.ua/bitstream/123456789/50877/1/%28167168%29_Kolisnichenko.pdf (дата звернення: 22.11.2023)

10. Коханевич Є.Г., Федюшин О.І Автоматизація аналізу безпеки програмного коду за допомогою платформи Kubernetes. Восьма міжнародна науково-технічна конференція «Проблеми інформатизації». Зб. Матеріалів форуму. Харків: ХНУРЕ. 2020. С. 95.
11. Полотай О.І. Ріст індексу розвитку економіки знань – основа ефективного управління освітніми проектами інформатизації. Управління проектами та розвиток виробництва: зб. наук. пр. Луганськ: Вид-во СНУ ім. В. Даля, 2012. № 3(43). С. 62-69.
12. Полотай О.І., Ноздріна Л.В. Internet-проекти, як базові інструменти управління знаннями в економіці. Нові інформаційні технології в освіті для всіх: стан та перспективи розвитку: зб. матер. 2-ої Міжнар. конф. К.: Вид-во IRTC, 2007. С. 392-398.
13. Проектні ризики і невизначеність. URL: https://www.oa.edu.ua/download/Lektsija_8.pdf (дата звернення: 22.11.2023)
14. Хто такий DevOps. URL: https://qagroup.com.ua/publications/khto-takyj- devops-engineer/ (дата звернення: 22.11.2023)
15. Чим займається DevOps – інженер. URL: https://dan-it.com.ua/uk/blog/kto-takoj-devops-inzhener-i-chem-on-zanimaetsja/#:~:text=DevOps%2D%D1%84%D0%B0%D1%85%D1%96%D0%B2
%D1%86%D1%96%20%D0%BF%D1%80%D0%B0%D1%86%D1%8E%D1%8E%D1%82%D1%8C%20%D0%BD%D0%B0%D0%B4%20%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%94%D1%8E,%D0%BD%D0%B0%D0%BF%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%2C%20%D0%BD%D0%BE%D0%B2%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BD%D0%B5%20%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F). (дата звернення: 22.11.2023)
16. Що таке DevSecOps? URL: https://web-academy.ua/blog/junior/what-is-devsecops (дата звернення: 22.11.2023)

17. Ящук В.І. Моделі вибору оптимального функціонування інформаційно- сервісних систем економічного об'єкту. Науковий вісник НЛТУ України: збірник науково-технічних праць. Львів: РВВ НЛТУ України, 2014. Вип. 24.4. С. 342-347.
18. Ящук В.І. Моделювання задач управління інвестиційними проектами в сфері гостинності в умовах ризику та невизначеності. Східна Європа: економіка, бізнес та управління. 2017. №4. URL: //www.easterneurope-ebm.in.ua/9-2017-ukr (дата звернення: 22.11.2023)
19. Ящук В. І. Онтологія наукових досліджень та методологія наукового пізнання. Економіка в контексті глобальних змін суспільства: матеріали Міжнародної науково-практичної конференції (м. Дніпро, 18 липня 2020 р.). Дніпро: НО «Перспектива», 2020. 140 с. (С.100-104).
20. Ящук В. І. Тренди використання технології «хмарних обчислень» в ІТ- сфері України. Торгівля, комерція, підприємництво : збірник наукових праць / [редакц. кол.: Апопій В. В., Дайновський Ю. А., Скибінський С. В. тощо.]. Львів: Львівська комерційна академія, 2012. Вип. 14. С. 104-108.
21. Ch.Schmittner, P. P. Puschner, Th. Gruber, Erwin Schoitsch. Security Application of Failure Mode and Effect Analysis (FMEA). ResearchGate. 2014. URL: https://www.researchgate.net/publication/290751391_Security_Application_of_Failure_Mode_and_Effect_Analysis_FMEA
22. DevSecOps: Інтеграція безпеки продукту на кожному етапі SDLC. URL: https://cloudfresh.com/ua/cloud-blog/devsecops-intehratsiya-produktu-bezpeky-na-kozhnomu-etapi-sdlc/ (дата звернення: 22.11.2023)
23. DevSecOps. Synopsys–2021. URL: https://www.synopsys.com/glossary/what-is-devsecops.html (дата звернення: 22.11.2023)
24. Failure Mode And Effects Analysis (FMEA) – How To Analyze Risks For Better Software Quality & Satisfied Customers! SOFTWARETESTINGHELP–2020. URL: https://www.softwaretestinghelp.com/failure-mode-and-effects-analysis-fmea/ (дата звернення: 22.11.2023)

25. Glenford J. Myers. The Art of Software Testing. 2011. 256 с.
26. Mulder J. Enterprise DevOps for Architects. Packt Publishing. BIRMINGHAM-MUMBAI, 2021. 178 с.
27. Nayan B. Ruparelia. Software development lifecycle models. International scientific conference. 2010. 6 с.
28. Nirali Honest. Role of Testing in Software Development Life Cycle. International Journal of Computer Sciences and Engineering. 2019. 6 с.
29. OWASP Top Ten. OWASP – 2020. URL: https://owasp.org/www-project- top-ten/ (дата звернення: 22.11.2023)
30. OWASP WebGoat. OWASP–2021. URL: https://owasp.org/www-project-webgoat/ (дата звернення: 22.11.2023)
31. PEARL XXI: PRISMA: Product Risk Assessment for Agile projects. Word Press.– 2014. URL: https://agilepearls.wordpress.com/tag/product-risk-management-prisma/ (дата звернення: 22.11.2023)
32. Project risk analysis and management. APM–2018. URL: https://www.apm.org.uk/media/10466/pram_web.pdf (дата звернення: 22.11.2023)
33. Risk Based Testing: Approach, Matrix, Process & Examples. Guru99. URL: https://www.guru99.com/risk-based- testing.html (дата звернення: 22.11.2023)
34. Static Application Security Testing. Synopsys. URL: https://www.synopsys.com/glossary/what-issast.html (дата звернення: 22.11.2023)
35. Understanding What is COBIT and COBIT Framework. Simplilearn – 2021. URL: https://www.simplilearn.com/what-is-cobit-significance-and-framework-rar309- article (дата звернення: 22.11.2023)
36. WHAT IS AGILE? WHAT IS SCRUM? CPRIME, INC. – 2021. URL: https://www.cprime.com/resources/what-is- agile-what-is-scrum/ (дата звернення: 22.11.2023)
37. What is DevOps?. AWS – 2020. URL: https://aws.amazon.com/devops/what-is-devops/ (дата звернення: 22.11.2023)
38. What is FaultTree Analysis. Sixsigmastudyguide. URL: https://sixsigmastudyguide.com/fault-tree-analysis/ (дата звернення: 22.11.2023)

39. What is risk management and why is it important? TechTarget– 2021. URL: https://searchcompliance.techtarget.com/definition/risk-management (дата звернення: 22.11.2023)
40. What is SABSA Enterprise Security Architecture and why should you care? Medium – 2019. URL: https://medium.com/@marioplatt/what-is-sabsa-enterprise-security-architecture-and- why-should-you-care-a649418b2742 (дата звернення: 22.11.2023)
41. What Is the MITRE ATT&CK Framework? McAfee – 2021. URL: https://www.mcafee.com/enterprise/en-us/security- awareness/cybersecurity/what-is- mitre-attack-framework.html (дата звернення: 22.11.2023)
42. Waterfall Methodology. Adobe – 2021. URL: https://www.workfront.com/project-management/methodologies/waterfall (дата звернення: 22.11.2023)
43. Welcome to Dynamics 365 Business Central. Microsoft–2021. URL: https://docs.microsoft.com/en-us/dynamics365/business-central/ (дата звернення: 22.11.2023)
44. 2021 CWE Most Important Hardware Weaknesses. Common Weakness Enumeration–2021. URL: https://cwe.mitre.org/ (дата звернення: 22.11.2023)