Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА
на тему:
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОРГАНІЗАЦІЇ ЗМАГАНЬ З КІБЕРБЕПЕЗКИ»

Виконав: студент 2 курсу, групи БС-21м
спеціальності 125 Кібербезпека
_____ Коркошко В. Р.

Керівник: к. т. н., доцент каф. ЗІ
_____ Куперштейн Л. М.
« 22 » грудня 2022 р.

Опонент: к. т. н. доцент каф. ПЗ
_____ Хошаба О. М.
« ___ » _____ 2022 р.

Допущено до захисту
Завідувач кафедри ЗІ
д. т. н., проф.
_____ Лужецький В. А.
«22» грудня 2022 р.

Вінниця ВНТУ – 2022 року

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації
Рівень вищої освіти II (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма – «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д. т. н., проф.
_____ В. А. Лужецький
15 09 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Коркошку Віктору Руслановичу

1. Тема роботи: «Інформаційна технологія організації змагань з кібербезпеки» керівник роботи: Куперштейн Леонід Михайлович, к. т. н., доцент, затверджені наказом ВНТУ №203 від 14.09.2022.
2. Строк подання студентом роботи – 19 грудня 2022 року
3. Вихідні дані до роботи:
   - засіб у вигляді веб-додатку, що складається з серверної та клієнтської частини;
   - засіб повинен дозволяти проводити змагання з кібербезпеки відповідно до інформаційної технології;
   - практичні завдання в категорії «веб» та «форензіка»
4. Зміст текстової частини: Вступ. 1. Аналіз предметної області. 2. Розробка інформаційної технології. 3. Програмна реалізація інформаційної технології. 4. Економічне обґрунтування. Висновки. Список використаних джерел. Додатки.
5. Перелік графічного матеріалу.
Схема процесів інформаційної технології (плакат, А4). Загальна модель відповідності завдань до ОПП «БІКС» (плакат, А4). Модель відповідності завдання категорії «форензіка» до ОПП «БІКС» (плакат, А4). Модель відповідності завдання категорії «веб» до ОПП «БІКС» (плакат, А4). Архітектура системи (плакат, А4). Вигляд головної сторінки змагання (плакат, А4).

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Куперштейн Л. М., доц. кафедри ЗІ		
2	Куперштейн Л. М., доц. кафедри ЗІ		
3	Куперштейн Л. М., доц. кафедри ЗІ		
4	Лесько О.Й., проф. кафедри ЕВПМ		

7. Дата видачі завдання – 1 вересня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітки
1	Аналіз завдання. Вступ	01.09.2022 – 04.09.2022	
2	Аналіз інформаційних джерел за напрямком магістерської кваліфікаційної роботи	05.09.2022 – 15.09.2022	
3	Науково-технічне обгрунтування	16.09.2022 – 22.09.2022	
4	Розробка технічного завдання	23.09.2022 – 04.10.2022	
5	Розробка рішень	05.10.2022 – 09.11.2022	
6	Практична реалізація, моделювання, експериментування, результати	10.11.2022 – 24.11.2022	
7	Аналіз виконання ТЗ, висновки	25.11.2022 – 29.11.2022	
8	Оформлення пояснювальної записки	30.11.2022 – 06.12.2022	
9	Попередній захист МКР та доопрацювання МКР	07.12.2022 – 19.12.2022	
10	Представлення МКР до захисту, рецензування	20.12.2022 – 21.12.2022	
11	Захист МКР	22.12.2022 – 26.12.2022	

Студент _____ (підпис) Коркошко В.

Керівник роботи _____ (підпис) Куперштейн Л. М.

АНОТАЦІЯ

УДК 004.855.5

Коркошко В. Р. Інформаційна технологія організації змагань з кібербезпеки. Магістерська кваліфікаційна робота зі спеціальності 125 – Кібербезпека, освітня програма – Безпека інформаційних і комунікаційних систем. Вінниця: ВНТУ, 2022. 93 с.

Укр. мовою. Бібліогр.: 37 назв; рис.: 45; табл.: 18.

Магістерська кваліфікаційна робота присвячена розробці інформаційної технології організації змагань з кібербезпеки разом з програмним засобом, що реалізує зазначену інформаційну технології.. В рамках роботи проведено аналіз підходів організації змагань з кібербезпеки, а також їх структуру. Виконано аналіз та порівняння існуючих рішень. Розроблено інформаційну технологію та відповідно програмний засіб. Розроблено практичні завдання в категорії «форензіка» та «веб». Виконано аналіз проведеного змагання.

Ілюстративна частина складається з 6 плакатів.

В економічному розділі оцінено витрати на розробку технології та програмного засобу.

Ключові слова: змагання, кібербепезка, захоплення прапорця;

ABSTRACT

Korkoshko V. R Information technology for organizing cyber security competitions. Master's thesis on specialty 125 – Cybersecurity, educational program - Security of information and communication systems. Vinnytsia: VNTU, 2022. – 93 p.

In Ukrainian. Bibliographer: 37 titles; fig.: 45; tabl.: 18.

The master's thesis is devoted to the development of information technology for the organization of cyber security competitions together with the software that implements the specified information technology. The work includes an analysis of approaches to the organization of cyber security competitions, as well as their structure. Analysis and comparison of existing solutions was performed. Information technology and, accordingly, a software tool have been developed. Practical challenges in the "forensics" and "web" categories have been developed. The analysis of the competition was carried out.

The illustrative part consists of 6 posters.

The economic section estimates the costs of technology and software development.

Keywords: ctf, competitions, cybersecurity, capture the flag;

ЗМІСТ

ВСТУП

На сьогоднішній час, використання інформаційних технологій є досить популярним та розповсюдженим, в результаті чого питання про надійну інформаційну безпеку стає актуальним з кожним днем. Для підготовки необхідних фахівців, що зможуть забезпечувати захист різного роду ресурсів у великої кількості навчальних-структур присутня спеціальність «кібербезпека». Вона в свою чергу займається вивченням різноманітних методів та практик захисту інформаційних систем від зловмисників.

Одним із важливих питань у підготовці майбутніх фахівців є те, що практичні навички в контексті пошуку та використання різного роду вразливостей і загроз є недостатніми. Оскільки, щоб виконувати та створювати якісний захист інформаційних систем потрібно розуміти, як насправді використовуються вразливості та як здійснюються різноманітні кібератаки.

Для вирішення цієї проблеми пропонується розборка інформаційної технології організації змагань з кібербезпеки, яка дозволить чинним і майбутнім фахівцям застосовувати свої знання на практиці на базі реальних систем.

Змагання в кібербезпеці зазвичай базуються на ідеї CTF (Capture the flag), що представляє собою командні змагання із захоплення прапора. Найбільш популярною реалізацією є "task-based" підхід, де команда учасників при виконанні завдання (наприклад, виявленні вразливості) отримує спеціальний прапор (як правило він має вигляд унікальної текстової мітки в контексті завдання), за яку команда отримує певну кількість в залежності від складності.

Об’єктом дослідження є процеси організації змагань з кібербезпеки.

Предметом дослідження є методи та засоби організації змагань з кібербезпеки.

Метою магістерської кваліфікаційної роботи є покращення умінь та навичок фахівців з кібербезпеки застосовувати знання у практичних ситуаціях за рахунок використання елементів гейміфікації.

Наукова новизна магістерської кваліфікаційної роботи полягає в створенні інформаційної технології організації змагань з кібербезпеки, що покращує здатність фахівців з кібербезпеки застосовувати знання у практичних ситуаціях за рахунок елементів гейміфікації та вкористанні реальних проблемних завдань, що відрізняється можливістю оцінювання рівня кваліфікації учасників змагань відповідно до ОПП зі спеціальності 125 «Кібербезпека» (перший рівень вищої освіти).

Практичною цінністю роботи є:

- практичні завдання категорій «веб» та «форензіка», розв’язання яких дозволяє покращити практичні навички із відповідних напрямків;
- програмний засіб у вигляді веб-додатку, що відповідає розробленій інформаційній технології організації змагань з кібербезпеки.

Результати роботи магістерської роботи доповідалися на L науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії Вінницького національного технічного університету (Вінниця, 2021 р.).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз підходів до організації змагань з кібербезпеки

На поточний момент найбільш розповсюдженим підходом в організації змагань з кібербезпеки є захоплення прапорця (capture the flag), де що представляє з себе командні або одиночні змагання із захоплення прапору. В процесі захоплення прапорців в якості винагороди учасники можуть отримувати спеціальні цифрові бали за виконання наданих завдань. В залежності від підходу учасники маються необхідність визначати потенційні загрози та виконувати повноцінний захист віддалених серверів від кібератак.

Існує декілька видів змагань, які проводяться в контексті CTF [2]:

1. Jeopardy (Task-based) – учасникам надається певний набір завдань (наприклад, для виявлення різного роду вразливостей) для яких потрібно знайти спеціальний прапорець. Зазвичай, прапорець маскується в спеціальну текстову мітку, наприклад, ZICTF{X…X}, де X – це будь–який символ в форматі ASCII. В результаті знаходження прапорця команді учасників нараховується певні бали, кількість яких відповідає складності виконаного завдання або безпосередньо його категорії. Як правило, контекст самих завдань є незалежний, тобто, вони між собою ніяким чином пов'язані. Відомі категорії завдань в змаганнях з кібербезпеки:

- Forensics – аналіз та відновлення різноманітних даних для з’ясування деталі подій, що відбулися в рамках кібератаки.
- Reversing – застосування навичок зворотного проектування, щоб дізнатись, що саме виконує програма.
- Pwn – пошук та експлуатація вразливостей в бінарних файлах.
- OSINT – аналіз загальнодоступних даних з цілю пошуку необхідної інформації.
- Stego – пошук прихованої інформації у файлах, картинка, тощо.

– Crypto – аналіз криптографічних функцій. Дешифрування різних об’єктів.
– Hardware – атаки на фізичні пристрої, що використовуються в сьогоденні.

2. Classic (Attack-Defense) – в даному типі змагання командам потрібно зосередитися безпосередньо на захисті та належному функціонуванні наданих віртуальних серверів чи мереж. У ході самого змагання учасники команд одержують спеціальни бали, але за підтримку працездатності системи, надійний захист, а також за проведення атак на ресурси інших команд.

3. King of the Hill (Король гори) – це вид змагання з кібербезпеки, в якому певна кількість команд змагається за контроль над цільовою системою. Основне завдання цього змагання – першим отримати доступ до системи та протидіяти отриманню доступу іншими командами.

4. Quiz – тест, котрий як правило містить в собі певну кількість питань та відповідей в області інформаційної безпеки.

1.2 Аналіз структури змагань з кібербезпеки

В якості відоми "PHDays-CTF", що в свій час проводилось на доволі якісному рівні, оскільки була створена командою "Positive Hack Days", котра займається проведенням конференцій з комп'ютерної безпеки. Перевагою "PHDays-CTF" є те, що всі вразливості не являються видуманими, присутні тільки ті, що дійсно зустрічаються в сучасних інформаційних системах [3].

1.2.1 Загальні положення

На "PHDays-CTF" кожна команда складалась з 3 - 5 учасників. Можливість заміни учасників виключно при погодженні з журі.

Також команди повинні були використовувати власні комп'ютерні пристрої. Формат прапорців був представлений у форматів MD5-рядків з довжиною в 32 символи.

Бали нараховувались за наступні досягнення:

– захоплення прапорців з сервісів противників;
– захоплення прапорців з загальних сервісів, які не належать окремим командам;
– перемогу в додаткових завданнях;
– в процесі утримання власних сервісів в рамках змагання "Король гори".

Бали вираховувались за наступні порушення:

– захоплення прапорів з власних сервісів суперниками;
– порушення доступності власних сервісів;
– порушення функціонування вразливих сервісів;
– порушення загальних правил у змаганнях.

В процесі змагання командам заборонялось:

– проводити кібератаки на комп'ютери, які відносять до журі;
– виконувати фільтрацію трафіку до будь-яких ресурсів;
– генерувати велику кількість трафіку (флуд);
– проводити деструктивні атаки на сервери інших;
– видаляти прапорці з серверів, на котрих розташовані вразливі додатки;
– виконувати маскування під іншу команду

Робота журі полягає в уточненні правил до початку змагання, видачі штрафів командам за порушення правил, а також в оголошенні переможців змагання по завершенню.

1.2.2 Інфраструктура CTF

На початку змагання команди отримують однакові сервери з встановленим набором вразливого програмного забезпечення. Головна задача кожного з учасників знайти вразливості, усунути їх та використати для отримання прапорців інших команд (противників).

Безпосередньо за процесом змагання постійно слідкує система журі, при цьому постійно змінюючи стан інфраструктури змагання (додаючи нові прапорці та вразливості).

Зі сторони організаторів заздалегідь готується обмежена кількість вразливих застосунків. Учасники працюють з двома типами систем:

– відкритими – маючі привілейований доступ на рівні операційної системи);
– закритими – маючі доступ виключно до вразливих сервісів виключно по мережі;

Додатково в рамках змагання “PHDays CTF” проводяться додаткові бонусні завдання, що направлені на розвагу учасників команд. Бонусні завдання можуть допомогти зібрати додаткові бали.

Для змагання учасникам пропонуються завдання наступних типів:

– виявлення та експлуатація вразливостей в сервісах команд суперників;
– виявлення та експлуатація вразливостей в загальних сервісах;
– «Король гори» з метою отримання та утримання контролю над сервісами;
– бонусні завдання.

Прапорці представлені у вигляді MD5 рядка з довжиною 32 символи, приклад наведено на рис 1.1.

Рисунок 1.1 – Приклад вмісту прапорця у вигляді текстового рядка MD5

Прапорець командних сервісів має наступні властивості:

– ідентифікатор команди (для уникнення передачі прапорців іншим командам);
– можливість захватити прапорець з сервісів інших команд;
– кожен прапорець може бути захоплений всіма командами, але тільки один раз.

В контексті властивостей прапорець загальних сервісів не містить в собі ідентифікатор команди. Також прапорці доступні для захоплення всім командам.

Нарахування балів відбувається відповідно до виду завдань і в принципі є індивідуальним для кожного організатора завдання. Для командних сервісів за кожен захоплений прапорець у команд суперників отримує, наприклад, 10 балів. При цьому діють штрафні бали, які знімаються за втрачений прапорець. Додатково штрафні бали нараховуються за порушення доступності своїх сервісів – 40 балів.

У випадку загальних сервісів (“task-based” підхід) команда може отримати бали залежно від категорії та складності завдання. Сумарна кількість балів в даному випадку являється індивідуальною для кожного змагання.

Також бали нараховуються і в конкурсі з підходом “король гори”, як правило організовується декілька видів сервісів в залежності від складності:

– 1 рівень складності;
– 2 рівень складності.

Для прикладу є доцільним розглянути хакатон “Vinnitsia Linuxation”, котрий в 2021 році проходив в контексті захисту інформаційних систем, що використовував такий підхід як “ King of the Hill” для подання завдань [4]. Було 2 рівня складності завдань:

1. Перший рівень був досить елементарним та полягав у підключенні до мережі VPN та наданого віртуального сервера використовуючи SSH. Далі потрібно було налаштувати підключення до S3-корзини AWS та завантажити завдання з наступним рівнем.

На другому рівні необхідно було забезпечити зовнішній доступ до стороннього серверу, доступ до якого був тільки з наданого віртуального серверу раніше. Наступним кроком потрібно було завантажити виконуваний файл, що дозволить виконання команд без посередньо на сервері. Після чого виконати модифікацію деяких скриптів та файлів на сервері, щоб досягнути змін на веб-сторінці, яка була розташована на окремій S3-корзині AWS [5].

Після проходження вказаних рівнів головною задачею було утримання своїх позицій, тобто потрібно було будь-яким чином запобігти модифікування критичних файлів командами суперниками.

Технічна сторона в організації таких змагань є досить індивідуальною і в загальному організатори можуть використовувати власні розробки, або безкоштовні та комерційні рішення в обмеженому варіанті. Більш детально це питання буде розглянуто в наступному розділі.

1.3 Аналіз існуючих рішень для проведення змагань з кібербезпеки

В рамках цього підрозділу необхідно розглянуто існуючі рішення, що дозволяють організовувати змагання з кібербезпеки, а також виконати їх порівняння за певними критеріями оцінки та визначити можливість їх використання в контексті інформаційної технології організації змагань з кібербезпеки.

В процесі пошуку рішень для організації змагань з кібербезпеки було виявлено певний дефіцит серед відкритих та актуальних проектів, що створені саме для організації змагань з кібербезпеки. Враховуючі цей факт, було вирішено також розглянути комерційні платформи, що надають можливість організовувати змагання з кібербезпеки. Результати пошуку відкритих платформ для проведення змагань з на Github наведено на рис. 1.2.

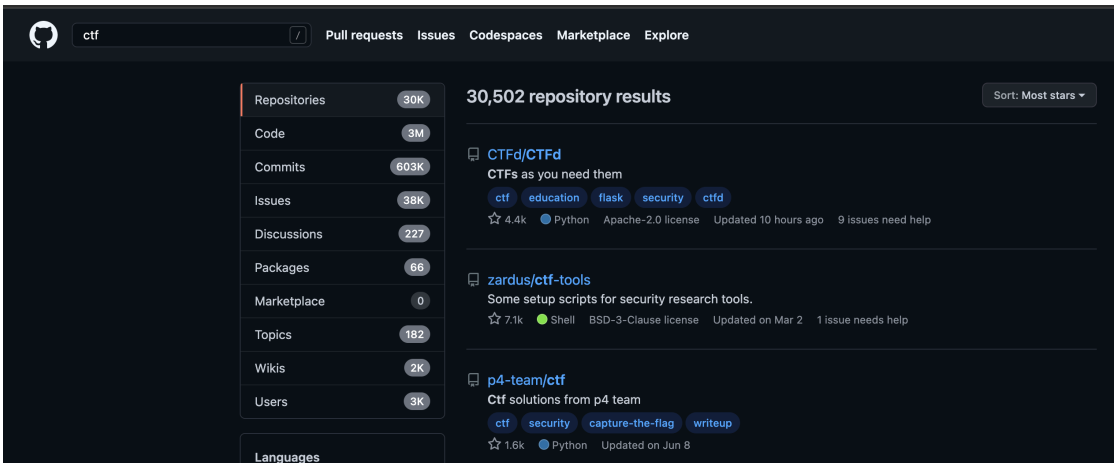


Рисунок 1.3 – Результат пошуку відкритих рішень на Github

Одним із відкритих рішень, що доступний на Github є проект CTFd, він має близько 4-х тисяч зірок та більше тисячі форків, що свідчить про привабливість цього проекту. CTFd – це фреймворк для проведення змагань з захоплення прапорця [6]. Зазначений фреймворк користується попитом серед організаторів змагань, оскільки це відкрите рішення з гарною документацію. Також, CTFd має додаткові платні функції та послуги, що в даному контексті може мати певні недоліки. Інтерфейс платформи CTFd зображено на рис. 1.4.

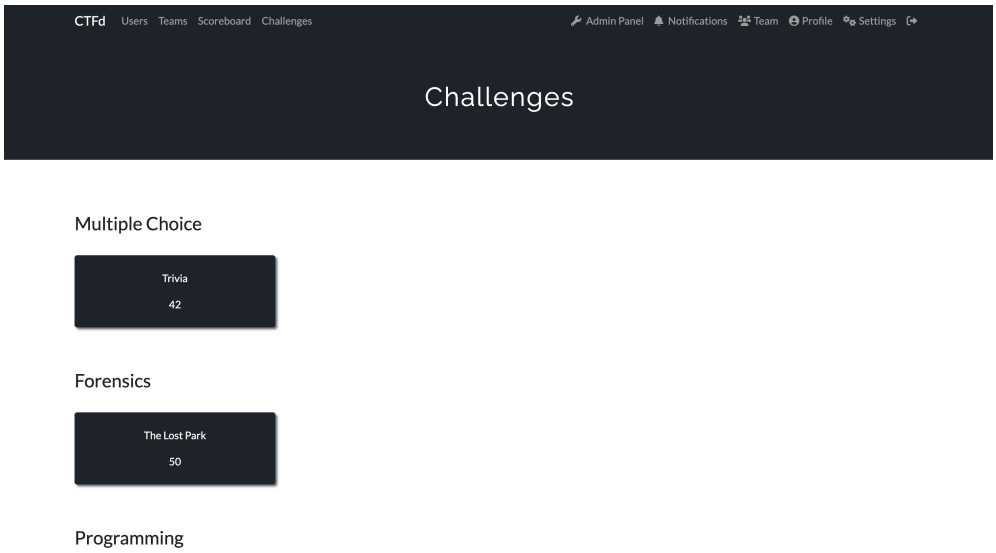


Рисунок 1.4 – Інтерфейс платформи CTFd

В якості закритого рішення для порівняння обрано досить популярну платформу HTB (Hack The Box) [7]. Він в свою чергу є комерційним проектом, що надає платні послуги з організації змагання на базі їх ресурсів. Інтерфейс платформи HTB зображено на рис. 1.5

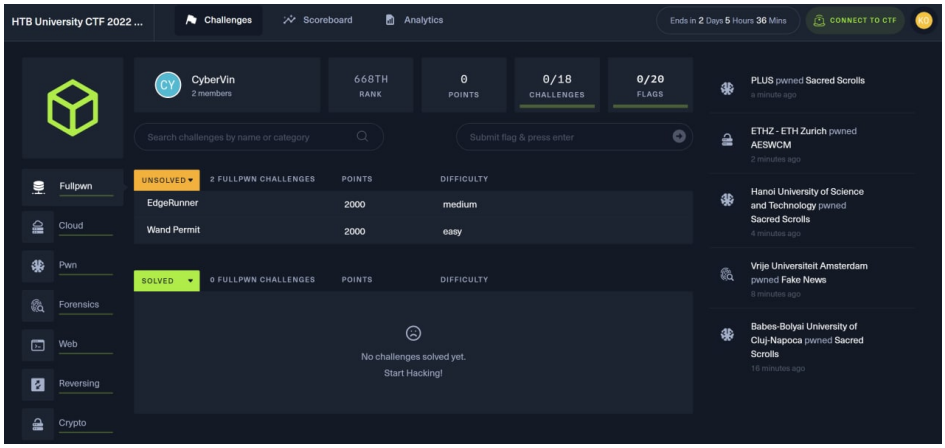


Рисунок 1.5 – Інтерфейс платформи HTB

Для проведення змагання на платформі HTB необхідно заповнити спеціальну форму на сайті для відправки запиту на розгляд та обговорення всіх деталей. Форма для відправки заявки її на розгляд зображена на рис. 1.6.

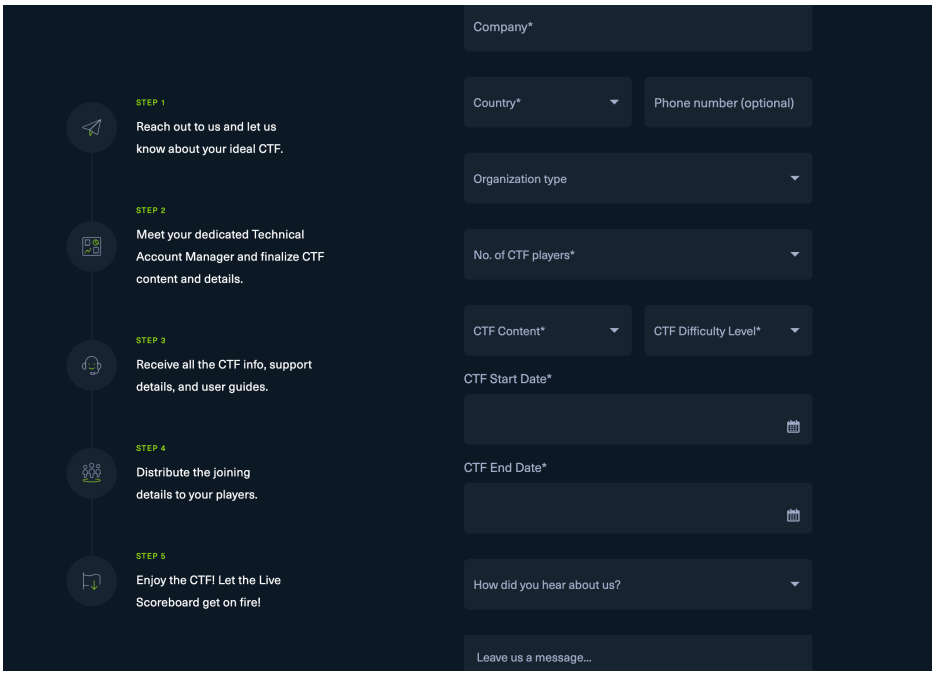


Рисунок 1.6 – Форма для відправки заявки на платформі HTB

Після загального розгляду знайдених кандидатів у вигляді CTFd та HTB виконаємо більше детальні порівняння між ними. Порівняння зазначених платформ наведено в таблиці 1.1.

Критерій оцінки	CTFd	HTB
Open-source	+	-
Self-hosted	+	-
Можливість модифікування системи	+	-
Проведення множини змагань одночасно	-	±
Автоматичне розгортання програмних завдань	± (в комерційній версії)	+
Документація	+	+
Зручність інтерфейсу	-	+

Таблиця 1.1 – Порівняння платформ CTFd та HTB

Відповідно до результатів порівняння, що зображені на таблиці 1.1 більш привабливою є платформа CTFd, оскільки вона підходить по 4.5 критеріям із 7.

В свою чергу HTB охоплює 3.5 критеріїв із 7 та є в певній мірі протилежністю CTFd. Хоча CTFd відповідно критеріям є більш найкращим варіантом, але його використання в якості реалізації інформаційної технології організації змагань з кібербезпеки не є можливим, оскільки платформа спроектована на рівні інтерфейсу та логіки під одноразове використання, а її доопрацювання рівносильно розробці нової платформи, шо не є доцільним та ефективним з точки зору складності та затрат часу.

Отже, в рамках даного підрозділу було проаналізовано існуючі рішення для проведення змагань з кібербезпеки. Також було виконано їх порівняння між собою за необхідними критеріями. Ні одне із рішень не підходить в повному обсязі відповідно до вимог інформаційної технології організації змагань з кібербезпеки.

1.4 Формалізація вимог та постановка задачі

В результаті проведеного аналізу можливих типів змагань з кібербезпеки, а також їх структури було визначено безпосередньо загальні принципи проведення такого роду змагань. Також, проаналізувавши існуючі рішення для організації змагань стало зрозуміло, що ні одне із існуючих рішень не підходить в повному обсязі у вигляді програмної частини для реалізації інформаційної технології.

Головною метою роботи є покращення умінь та навичок фахівців з кібербезпеки застосовувати знання у практичних ситуаціях за рахунок використання елементів гейміфікації.

Відповідно до цього сформовано список задач, які необхідно вирішити:

- описати існуючі процеси в рамках інформаційної технології;
- розробити практичні завдання для змагання в категоріяї «веб» та «форензіка»;
- розробити архітектуру програмного засобу, що відповідає інформаційній технології;

- визначити необхідні інструменти для реалізації програмної частини інформаційної технології;
- розробити програмну частину завдань в категорії «веб» та «форензіка».
- розробити серверну частину для проведення змагань;
- розробити клієнтські частини для організатора та учасника;
- проаналізувати результати проведеного змагання.

Отже, в рамках даного підрозділу сформовано та описано задачі, які необхідно виконати в рамках магістерської кваліфікаційної роботи.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

2.1 Структура інформаційної технології

Інформаційна технологія організації змагань з кібербезпеки складається з певних послідовних процесів. В контексті інформаційної технології, що розробляється виділено дві групи процесів:

1) Процеси на рівні організатора.
2) Процеси на рівні учасника.

Загальні схеми зазначених процесів інформаційної технології наведено на рис. 2.1.

Рисунок 2.1 – Схеми процесів інформаційної технології а) на рівні організатора б) на рівні учасника

Опис процесів на рівні організатора.

1) Процес підготовки. Процес містить в собі одні з найважливіших етапів:
   – дослідження існуючих вразливостей в контексті задач кібербезпеки. Для прикладу одним із основних джерел можна використовувати CVE [8]. CVE – це спеціальна база даних відомих вразливостей у сфері інформаційної безпеки. Кожна вразливість ідентифікується за допомогою унікального коду у форматі “CVE-рік-номер”, наприклад, CVE-2021-44228. Зазначена база даних вразливостей є досить поширеною та використовуються у великої кількості спеціалізованих сканерів, що займаються пошуком наявності можливих вразливостей у різного роду інформаційних систем.

– практичний розбір виконаних досліджень. Наприклад, відображення атаки в тестовому середовищі, що підтверджує існування вразливості для подальшого використання її у процесі створення завдання;
– створення сценаріїв. Даний етап відповідає за створення певної легенди в контексті змагання, щоб зробити більш цікавим і різноманітним процес зі сторони учасника. Враховуючи поточну ситуацію в країні для прикладу можна використати контекст протистояння з так званими “орками”.

Схема процесу підготовки на рівні організатора зображена на рис. 2.2.

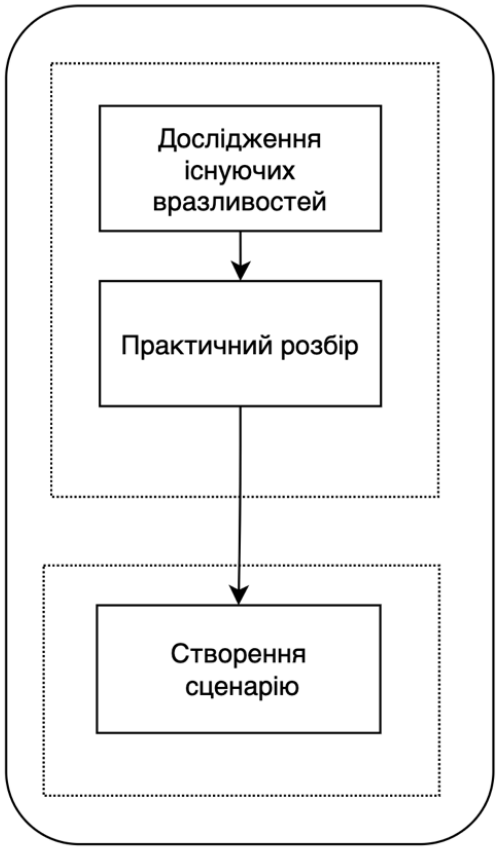


Рисунок 2.2 – Схема процесу підготовки на рівні організатора

2) Процес створення завдання. Процес складається з двох етапів:
– реалізації та формування завдання на базі результатів процесу підготовки. Включає в себе реалізацію завдання відповідно до контексту сценарію та виконаних досліджень по існуючим вразливостям.

– тестування на адекватність. Відповідає за перевірку розроблених завдань попереднього кроку “реалізації та формування завдань” на реалістичність та можливість їх виконання потенційним учасником змагання, котрий як правило являється чинним або майбутнім фахівцем з кібербезпеки.

Схема процесу створення завдання на рівні організатора зображена на рис. 2.2.

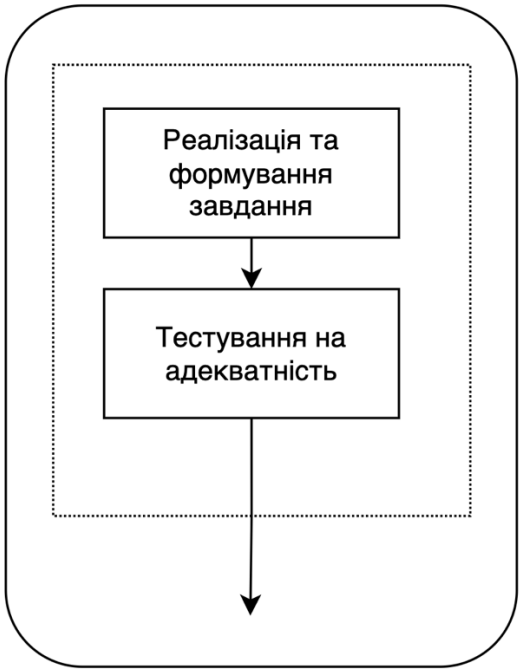


Рисунок 2.3 – Схема процесу створення завдань на рівні організатора

3) Процес проведення змагання. Вказаний процес є основним і складається з наступних етапів:
   – запуск змагання. На даному етапі відповідно до його назви виконується запуск змагання для учасників що зареєструвались та сформувались в команди. Також включає в себе розроблені завдання з попереднього процесу “створення завдань”, що демонструє чітку їх чітку послідовність.
   – моніторинг системи. Включає в себе аналіз стану змагання та збір даних про події, що відбуваються, наприклад, вирішення завдань, невдалі спроби, використання підказок, активність учасників, тощо.

– підтримка учасників. На поточному етапі організатори надають технічну підтримку учасникам, які стикаються з різноманітними проблемами, що можуть бути пов'язаними з неможливістю входу до власного кабінету системи або отриманню доступу до ресурсів, що є необхідними для виконання завдань. Також, відповідно до моніторингу організатори можуть аналізувати стан змагання, бачити проблемні задачі та відповідно до цього приймати рішення про публікацію спеціальних підказок, що можуть бути використані учасником. Це дозволяє додати більше контексту учасникам, щоб надихнути їх на подальші спроби виконання завдань у випадку певних складнощів та збільшити вірогідність успіху на позитивний результат.

Схема процесу проведення змагання на рівні організатора зображена на рис. 2.4.

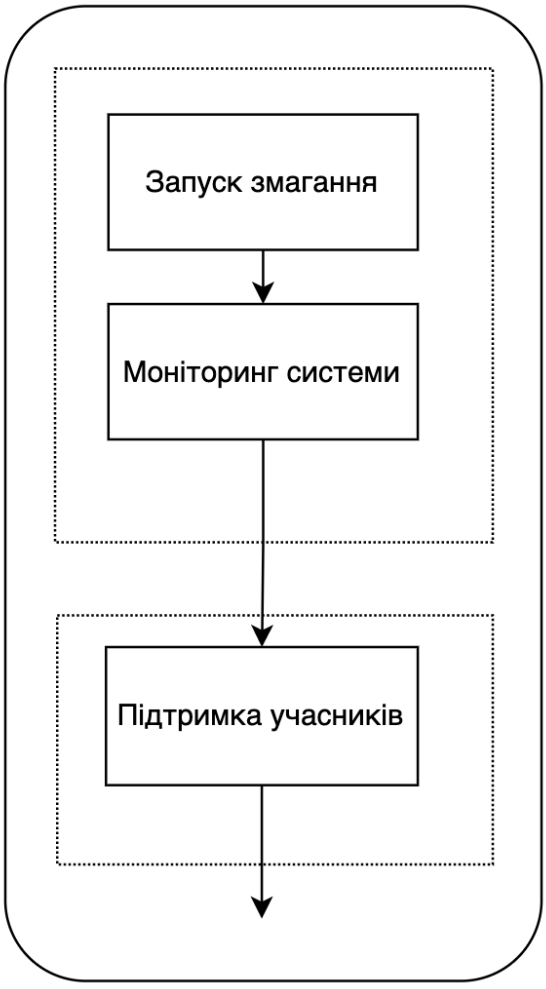


Рисунок 2.4 – Схема процесу проведенню змагання на рівні організатора

4) Процес аналізу результатів. Складається з наступних етапів:
   - аналіз зібраних даних. На даному етапі виконується аналіз зібраних даних на базі яких можна оцінити ефективність змагання та потенційний рівень учасників відповідно до областей кібербезпеки (залежить від різноманітності завдань). Також, на базі зібраних даних визначаються безпосередньо переможці змагання.
   - розбір рішень. Даний етап є опціональним в залежності від мотиву проведеного змагання. Всім учасникам відкривається доступ до докладно описаних кроків виконання завдань, що використовувались у змаганні. Це може допомогти учасникам проаналізувати свої помилки у підході до виконання завдань та підвищити свої здібності як фахівця з кібербезпеки.
   - висновки. Відповідно до аналізу зібраних даних формуються заключні висновки про змагання та виконується його офіційне закриття з зафіксованими переможцями.

Схема процесу аналізу результатів на рівні організатора зображена на рис. 2.5.

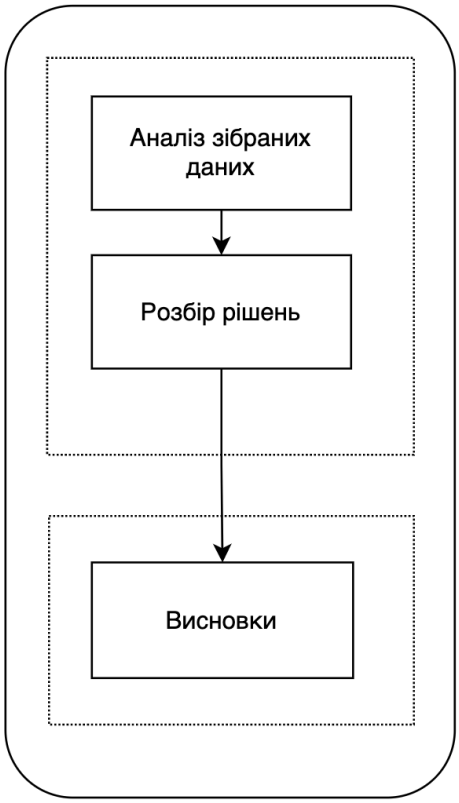


Рисунок 2.5 – Схема процесу аналізу результатів на рівні організатора

Розгляд процесів на рівні учасника.

1) Процес реєстрації учасника. Складається з наступних етапів:
   – реєстрація в системі. Учасник реєструється в системі за загальним або приватним посиланням.
   – створення команди. Даний крок є опційним. Учасник створює команду з унікальною назвою.
   – запрошення учасників до команди. Даний крок є опційним. Учасник запрошує потенційних учасників до створеної команди.

   Схема процесу реєстрації зображена на рис. 2.6.

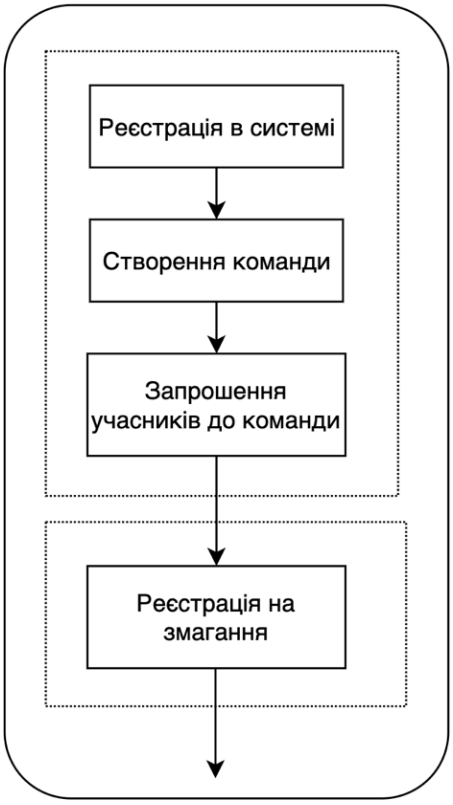


Рисунок 2.6 – Схема процесу реєстрації на рівні учасника

2) Процес знайомлення. Складається з наступних етапів:
   – вибір категорії завдань. На даному етапі учасник обирає цікаву йому категорію завдань для подальшого ознайомлення.
   – вибір завдання. Учасник обирає доступні завдання з обраної категорії.
   – ознайомлення з вимогами завдання.

   Схема процесу ознайомлення на рівні учасника зображена на рис. 2.7.

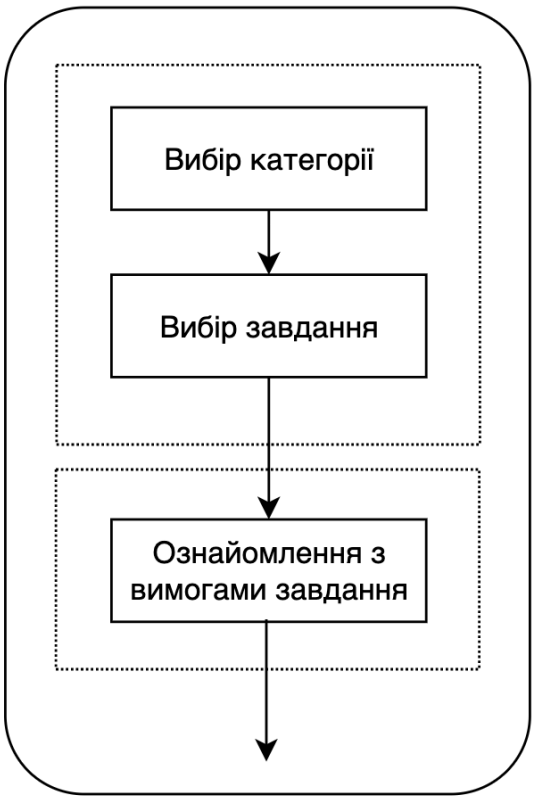


Рисунок 2.7 – Схема процесу ознайомлення на рівні учасника

3) Процес виконання завдання. Процес складається з наступних етапів:
   – ознайомлення з наданими ресурсами обраного завдання. Ресурс представляє з себе дані про віддалений сервер або спеціальні файли для завантаження, що необхідні для виконання завдання.
   – пошук вразливих місць. В результаті ознайомлення з ресурсами учасник виконує пошук слабких місць в контексті завдання.
   – проведення атаки. Відповідно до знайдених вразливих місць або припущенні певних гіпотез учасник здійснює атаку на ресурс. Атака може містити в собі певний ланцюг дій, що означає необхідність учаснику виконати декілька різноманітних атак у певній послідовності для досягнення необхідного результату.
   – отримання прапорця. В результаті успішної атаки на ресурсу в попередньо етапі учасник отримує відповідне підтвердження у вигляді прапорця, за який в подальшому отримує певну кількість очок.

Схема процесу виконання завдання на рівні учасника зображена на рис. 2.8.

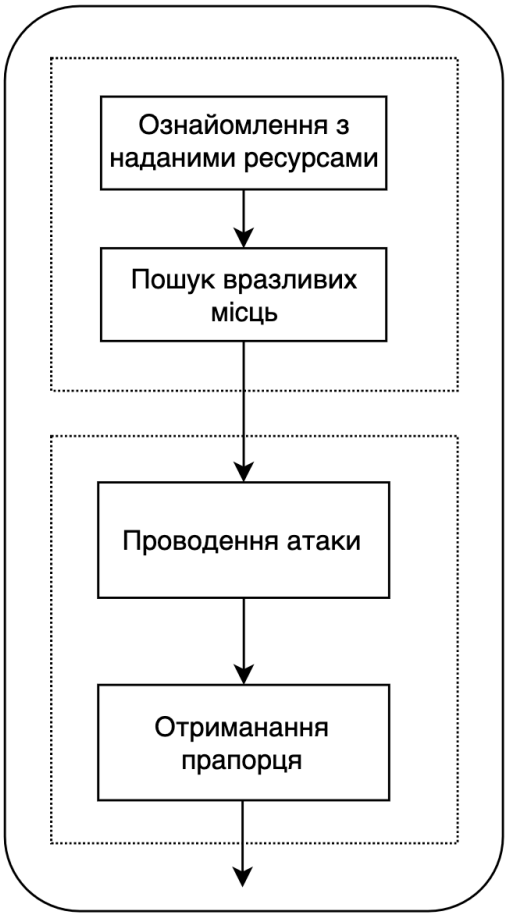


Рисунок 2.8 – Схема процесу виконання завдання на рівні учасника

4) Процес ознайомлення з результатами. Складається з наступних етапів:
   – отримання результатів. На даному етапі учасник має можливість ознайомитись з результатами змагання. Наприклад, переглянути фінальний топ команд, ознайомитись з завданнями які були виконанні найбільш виконані та навпаки.
   – отримання рішень. На даному етапі учасник може отримати докладний опис рішення всіх завдань (опційно).
   – отримання рекомендацій. Відповідно до результатів виконаних завдань учасник може отримати певні рекомендації з літератури для покращення знань з проблемних областей, що визначаються системою.

Схема процесу ознайомлення з результатами на рівні учасника зображена на рис. 2.9.

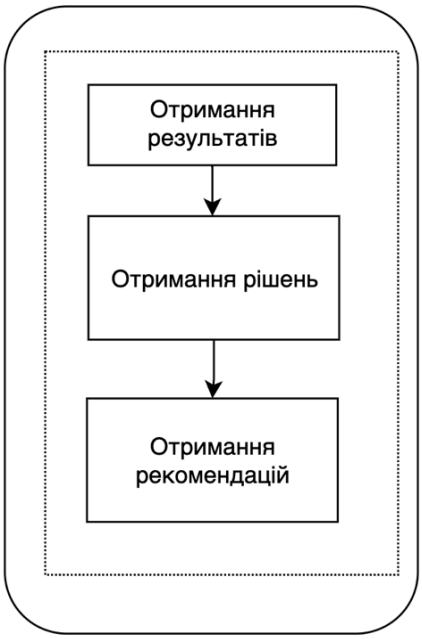


Рисунок 2.9 – Схема процесу ознайомлення з результатами на рівні учасника

Також в контексті проведення змагань в навчальному процесі пропонується підхід зі створюванням завдань відповідно до освітньо-професійної програми безпека інформаційних та комунікаційних систем (в подальшому ОПП «БІКС») [9]. Такий підхід може дозволити детальніше аналізувати результати змагання та бачити проблематику результатів навчання з відношенням до обо’язкових освітніх компонентів.

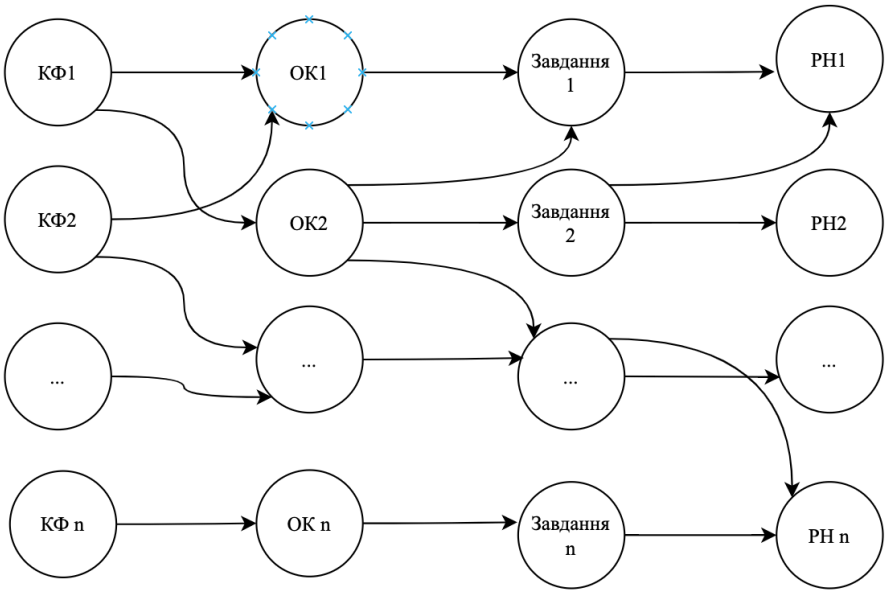


Рисунок 2.10 – Узагальнена модель відповідності завдань до ОПП «БІКС» (бакалавр)

На рис. 2.10 зображена узагальнена модель відповідності завдань до ОПП «БІКС». Присутні такі значення на рис. 2.10:

– КФ – фахові компетентності ОПП «БІКС»;
– ОК – освітній компонент ОПП «БІКС»;
– РН – результат навчання в рамках ОПП «БІКС».

2.2 Розробка практичних завдань для змагання

Однією з важливих практичних цінностей інформаційної технології є розробка практичних завдань в контексті задач кібербезпеки. В якості демонстрації методів розробки завдань на рівні організатора є доцільним розробити декілька завдань з різних категорії. Для цієї задачі обрано напрямки криміналістики (forensics) та веб в причину своєї популярності у сучасному світі IT-систем.

2.2.1 Розробка завдання з категорії «форензіка»

За своєю природою завдання у напрямку криміналістики пов’язані з аналізом різних файлів (вихідні файли, бінарні файли, журнали, тощо), щоб встановити ланцюжок події та дізнатись за допомогою яких вразливостей була виконана атака на ресурс. Безперечно практичні навички у криміналістиці є досить важливими для фахівця з кібербезпеки, тому акцент у розробці завдань для даного напрямку вважається доцільним.

Джерелом для завдань в області криміналістики можна обрати базу даних CVE. Досить зручним веб-застосунком, що оперує даними про вразливості та їх оцінками рівня критичності є “CVE Details” [10]. Тому скористаємось ним та переглянемо список вразливостей за минулий з фільтром високої критичності. Результат пошуку наведено на рис. 2.10.

Рисунок 2.10 – Результат пошуку критичних вразливостей

Після ознайомлення з результатами пошуку базуючись на популярності продукту була обрана вразливість під номером CVE-2021-44228, яка відноситься до бібліотеки Apache Log4j для мови програмування Java. Вказана бібліотека використовуються для введення журналу в рамках програмного забезпечення (наприклад, запис різних помилок або подій) та користується попитом серед розробників (3 тисячі зірок на github). Дана вразливість достатньо легко експлуатується та дозволяє виконувати довільний код зловмиснику, що може призвести до повного контролю над сервером. Експлуатація відбувається шляхом відправки зловмисником спеціальної комбінації, що в подальшому записується до журналу. Наприклад, в рамках веб-додатків на Java може бути реалізований журнал веб-запитів за допомогою зазначеної бібліотеки. Таким чином, зловмиснику достатньо помістити вразливу комбінації, наприклад, в якості значення HTTP-заголовку та відправити його на цільовий сервер. Приклад вразливої комбінації для експлуатації CVE-2021-44228 наведено на рис. 2.11.

```
X-Api-Version: ${jndi:ldap://zi.vntu.edu.ua/resource}
```

Рисунок 2.11 – Приклад вразливої комбінацій в якості значення HTTP-заголовку для експлуатації CVE-2021-44228

Виходячи з інформації про експлуатацію обраної вразливості за допомогою HTTP-запитів, доцільно створити, наприклад, на базі тестової локальної мережі дамп трафіку, в котрому будуть зафіксовані спроби експлуатувати зазначену вразливість. Це в свою чергу відображає концепт криміналістики, оскільки головною задачею учасника буде аналіз записаного дампу, наприклад, за допомогою інструменту Wireshark та пошук шкідливого запиту [11].

Покроковий алгоритм дій створення завдання:

1) Дослідження та аналіз джерел. В результаті даного кроку є знайдена загально відома та досить критична вразливість з ідентифікатором CVE-2021-44228, що дозволяє зловмиснику виконувати будь-який код в рамках цільового програмного забезпечення.
2) Практичний розбір.
3) Створення сценарії. Учаснику необхідно виконати ряд наступних дій для виконання завдання:

– завантаження дампу трафіка та відкриття його за допомогою зручного інструмента. Пропонується Wireshark;
– аналіз пакетів з виконанням фільтрації для вилучення лишніх результатів;
– знаходження пакету, що містить в собі шкідливу комбінацію;
– зіставлення шкідливої комбінації з існуючою CVE (в даному випадку прапорцем є ідентифікатор CVE).

4) Формалізація завдання. На даному кроці важливо описати завдання та його цілі для учасника.

5) Розробка завдання та тестування на адекватність. Результати даного кроку будуть описані в розділі з програмної реалізації системи та задач до неї.

Ця задача гарно перевіряє результати навчання освітнього компонента “інформаційно-комунікаційні системи”. Схема відповідності завдання з категорії криміналістики до ОПП 125 кібербезпека зображено на рис. 2.12.

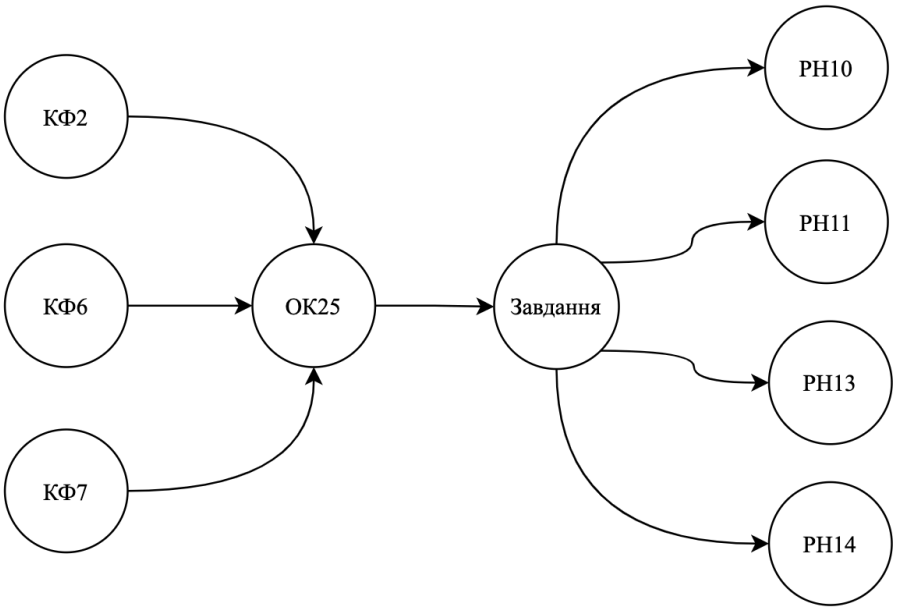


Рисунок 2.12 – Схема відповідності завдання з категорії «форензіка» до ОПП «БІКС»

На рис. 2.12 присутні таки позначення:

– ОК25 – це освітня компонента «Інформаційно-комунікаційні системи» в ОПП «БІКС»;
– РН10 – результат навчання №10 в ОПП «БІКС»: вміння виконувати аналіз та декомпозицію інформаційно-телекомунікаційних систем;
– РН11 – результат навчання №11 в ОПП «БІКС»: вміння виконувати аналіз зв’язків між інформаційними процесами на віддалених обчислювальних системах;
– РН13 – вміння аналізувати проекти інформаційно-телекомунікаційних систем базуючись на стандартизованих технологіях та протоколах передачі даних;
– РН14 – вміння вирішувати завдання захисту програм та інформації, що обробляється в інформаційно-телекомунікаційних системах програмно-

апаратними засобами та давати оцінку результативності якості прийнятих рішень.

2.2.2 Розробка завдання категорії «веб»

Як правило, завдання в контексті веб-додатків в більшості випадків пов’язані з неохайністю розробників при створенні нових систем та при сліпому використанні різноманітних інструментів без повного розуміння їх роботи. В рамках поточного підпункту в якості джерела інформації про загрози було обрано ресурс “Walarm” [12]. Walarm – це компанія, що спеціалізується на захисті веб-систем в контексті API. На їхньому сайті є велика кількість статей в різних напрямках, де, наприклад, описуються можливі вразливості та деталізується процес їх виконання. Тому, скористаємся цим та переглянемо доступні статі в категорії “API Security”, що відповідає типу розроблюваного завдання.

В процесі аналізу їх матеріалів більш цікавою виявилась стаття про “GraphQL Batching Attack” [13]. В ній описується можливість зловживання пакетними запитами в технології GraphQL [14].

GraphQL – це спеціальна мова запитів для API, що в свій час була розроблена компанією Facebook. Приклад пакетного запиту наведено на рис. 2.13. В даному прикладі зображено три мутаційні запити, кожне з яких викликає мутацію поля “newNumber” у об’єкта.

```
{
  first: changeTheNumber(newNumber: 1) {
    theNumber
  }
  second: changeTheNumber(newNumber: 3) {
    theNumber
  }
  third: changeTheNumber(newNumber: 2) {
    theNumber
  }
}
```

Рисунок 2.13 – Приклад пакетного запиту в GraphQL

Пакетні запити з першу виглядають досить зручними та корисними, але це завжди викликає питання та нюанси з точки зору безпеки. Наприклад, це може створювати проблеми у випадку використання одноразових паролів, що також відомі як OTP (one-time password) для двофакторної аутентифікації [15]. Тобто, зловмисник може виконати атаку методом грубого перебору для зламу одноразового паролю, передаючи десятки або сотні мутацій в рамках одного запиту до веб-сервера. Додатково це дозволяє обходити різні обмеження швидкості зі сторони веб-сервера.

Тому, базуючись на знайденій інформації є доречним створити завдання у вигляді веб-додатку, що реалізує функцію двофакторного підтвердження при аутентифікації користувача з використанням GraphQL. В якості мови програмування рекомендується використати JavaScript на платформі Node.JS [16]. Node.js - дозволяє створювати серверні веб-додатки. Основною перевагою рекомендованої мови є простота, швидкість та наявність офіційної та якісного імплементації GraphQL.

Головною задачею для учасника буде підібрати одноразовий пароль для двофакторного підтвердження. В якості додаткової складності та виключення можливості підбору одноразового паролю стандартними шляхами необхідно налаштувати певні обмеження швидкості на стороні веб-сервера.

Покроковий алгоритм дій створення завдання:

1) Дослідження та аналіз джерел. Результатом цього кроку є знайдена можливість зловживання пакетними запитами GraphQL, котра також відома як “GraphQL Batching Attack” та може використовуватись для обходу обмежень швидкості та задач грубого перебору у випадках з двофакторим підтвердженням.
2) Практичний розбір.
3) Створення сценарію. Учаснику необхідно виконати ряд наступних цій для вирішення завдання:
   – аналіз технологій, що використовує веб-додаток;
   – вивчення запитів, що відправляються клієнтською частиною до серверу;

– пошук додаткової інформації та слабких місць в рамках отриманої інформації з попередніх дій;
– спроби виконати атаку у разі знайдення інформації про пакетні запити в GraphQL;
– увійти до системи та отримати прапорець.

4) Формалізація завдання. На даному кроці є важливим описати завдання для учасника. В рамках поточних обставин пропонується наступний опис:

«Агентурній мережі ельфів вдалось перехопити деякі письмена орків, за якими стало зрозуміло, що вони планують секретну спеціальну операцію по загарбництву територій ельфів. Додатково вдалось дізнатись, що карта з детальним планом нападу орків з ціллю маскування зберігається на сайті компанії, що займається виробництвом пакетів. Нам також вдалось отримати дані для входу в кабінет управління, але там присутнє двофакторне підтвердження. Допоможіть спасти ельфів та отримайте доступ до кабінету.»

5) Розробка завдання та тестування на адекватність. Результати даного кроку будуть описані в розділі з програмної реалізації системи та задач до неї.

Модель відповідності завдання категорії «веб» до ОПП 125 кібербезпека зображена на рис. 2.14.

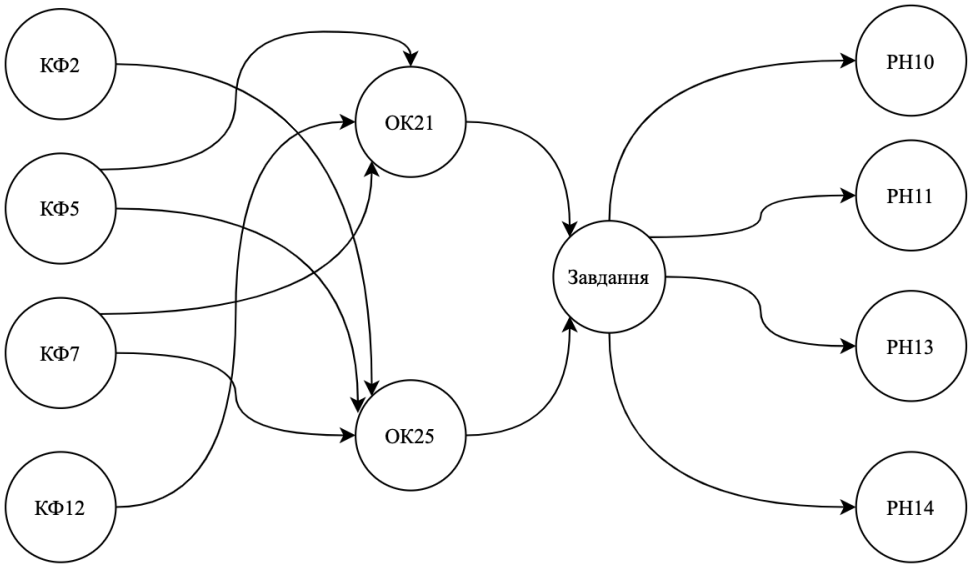


Рисунок 2.14 – Модель відповідності завдання категорії «веб» до ОПП «БІКС»

На рис. 2.14 присутні такі позначення:

- ОК25 – це освітня компонента «Інформаційно-комунікаційні системи» в ОПП «БІКС»;
- ОК21 – це освітня компонента «Програмування» в ОПП «БІКС»;
- РН10 – результат навчання №10 в ОПП «БІКС»: вміння виконувати аналіз та декомпозицію інформаційно-телекомунікаційних систем;
- РН11 – результат навчання №11 в ОПП «БІКС»: вміння виконувати аналіз зв'язків між інформаційними процесами на віддалених обчислювальних системах;
- РН13 – вміння аналізувати проекти інформаційно-телекомунікаційних систем базуючись на стандартизованих технологіях та протоколах передачі даних;
- РН14 – вміння вирішувати завдання захисту програм та інформації, що обробляється в інформаційно-телекомунікаційних системах програмно-апаратними засобами та давати оцінку результативності якості прийнятих рішень.

Отже, в рамках даного підрозділу розроблені практичні завдання в рамках категорії «форензіка» та «веб». Сформульовано покроковий алгоритм для кожного з завдань. Побудована модель відповідності до ОПП 125 – кібербезпека.

2.3 Архітектура програмного засобу для організації змагань

Система, що реалізує інформаційну технологію повинна представляти з себе веб-додаток, за допомогою якого можна проходити описані процеси у підрозділі 2.1.

Відповідно до завдання, система повинна складатись з клієнтської та серверної частини, що представляють з себе розділені програмні засоби. Такий підхід відповідає сучасним тенденціям у веб-розробці та дозволяє зменшити навантаження для розробників, які можуть в майбутньому приєднатись до підтримки системи і фокусуватись виключно на своїй спеціалізації. Також, в

якості переваги отримуємо незалежність серверної частини від клієнтської, що дозволить без проблем вдосконалювати інтерфейс.

В результаті вказаної інформації та вимог до системи, що реалізовує інформаційну технологію організації змагань з кібербезпеки побудовано архітектуру, що зображена на рис. 2.15.

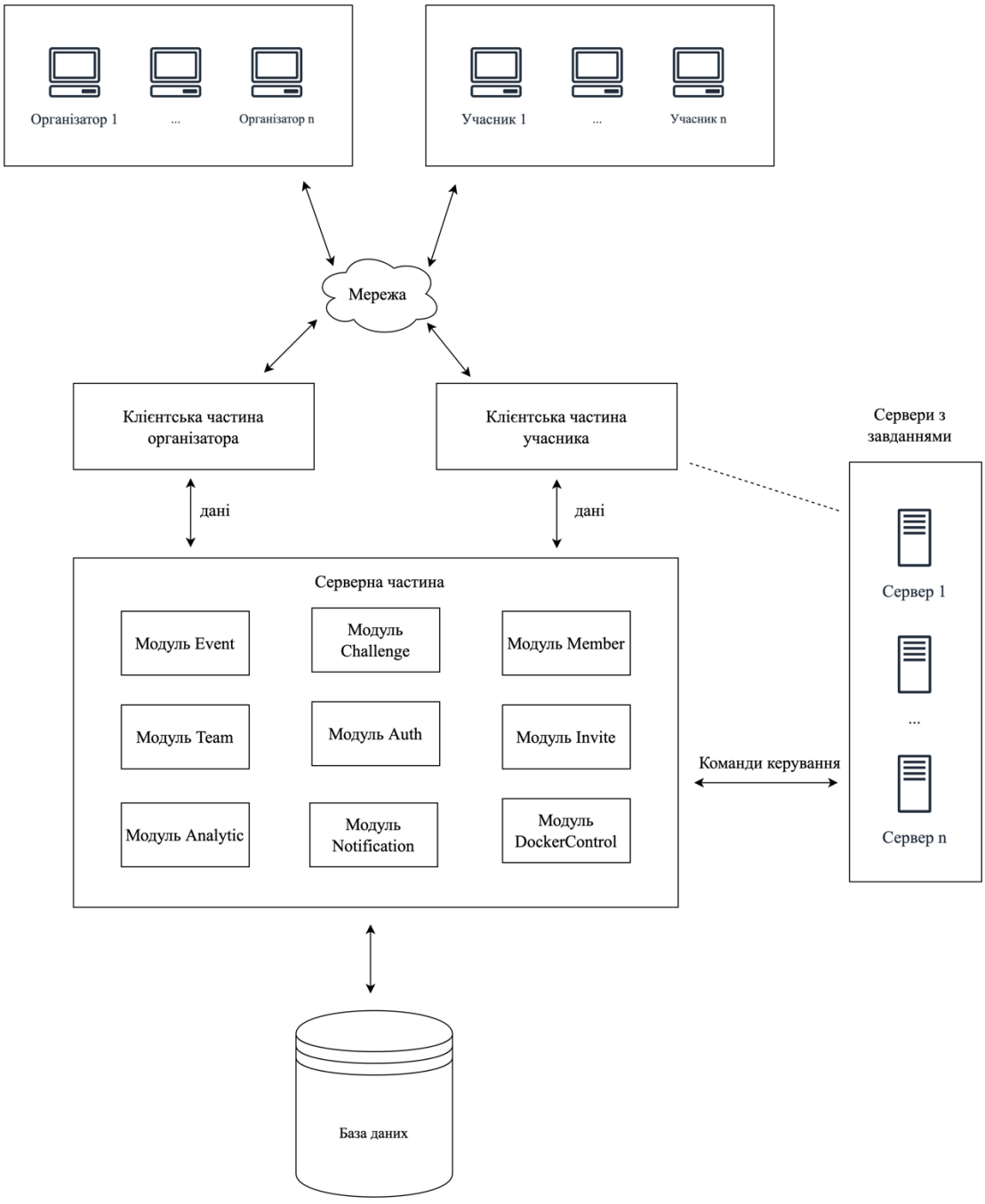


Рисунок 2.13 – Архітектура системи для організації змагань з кібербезпеки

Клієнтська частина організатора – веб-інтерфейс за допомогою якого організатор виконує наступні функції:

- створення, редагування, запуск та зупинка змагань;
- додавання та редагування завдань в контексті змагання;
- перегляд аналітики по змаганням;
- керування командами та учасниками;

Клієнтська частина учасника – веб-інтерфейс за допомогою якої учасник може виконувати наступні процеси:

- реєстрація та вхід до системи;
- створення команди;
- запрошення учасників до команди;
- перегляд доступних змагань та реєстрація на змагання;
- ознайомлення з завданнями та отримання ресурсів для їх виконання;
- перегляд рахункової таблиці та учасників змагання;

Серверна частина – це веб-додаток з реалізацією REST API [17]. Саме за допомогою API клієнтські частини організатора та учасники виконують процес комунікації з серверною частиною для виконання різних дій та отримання різних даних. Відповідно до процесів інформаційної технології серверну частину спроектовано у вигляді окремих модулів:

- Event – відповідає за створення, запуску та завершення змагання. Модуль повинен підтримувати можливість проведення паралельних змагань.
- Challenge – відповідає за логіку управління завданням та їх використанням у змаганнях.
- Member – відповідає за логіку учасника в системі. Зберігає базову інформацію про нього;
- Team – відповідає за логіку формування команд в системі;
- Invite – логіка запрошення нових учасників. Використовується модулем Team.
- Auth – аутентифікація та авторизація учасника або організатора в рамках системи.

- Analytic – збір даних та побудова на їх основи відповідної аналітики по змаганням.
- Notification – повідомлення учасників про нові події в контексті системи або змагання;
- Attachment – завантаження файлів в систему. Використовується модулями Event та Challenge.
- DockerControl – модуль керування контейнерами за допомогою інструменту Docker, що можуть містити в собі різноманітне програмне забезпечення [18]. Дозволяє запускати під кожну команду індивідуальний екземпляр вразливого ресурсу. Це вирішує проблему конфлікту при виконанні завдання багатьма учасниками одночасно. Також стає доступною можливість скидання стану наданого ресурсу у випадку виникнення різноманітних проблем в результаті дій учасника.

Отже, в рамках даного підрозділу описано основі складові системи у вигляді модулів та відповідно до цього спроектована архітектура системи відповідно до вимог.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ

3.1 Обґрунтування вибору інструментальних засобів розробки

Поточним етапом необхідно для програмної реалізації інформаційної технології організації змагань з кібербезпеки обрати інструменти, що будуть використовуватись в процесі розробки.

Для реалізації серверної частини платформи обрано мову програмування PHP [19]. Вона гарно підходить для веб-додатків та є досить сучасної з набор гарних практик, по типу стандартів PSR [20]. Однієї з переваг в екосистемі PHP є пакетний менеджер Composer що дозволяє з легкістю керувати залежностями проекту [21]. Додатково пакетний менеджер має потужну базу розширень, наприклад, для сканування залежностей проекту на наявність загальновідомих вразливостей. Також, у світі PHP існує значна кількість фреймворків, але монополістами і найпотужнішими являються Symfony та Laravel, тому залишається тільки вибір між ними двома. Головним фактором у відборі фреймворку є спрощеність та велика підтримка зі сторони спільноти.

Розглядаючи Symfony можна зробити певний висновок, що він в певному обсязі диктує структуру додатку. При цьому у Symfony достатньо потужна та велика екосистема з компонентів, що представлені у вигляді бібліотек [22]. В свою чергу Laravel працює відразу з моменту ініціалізації та не потребує специфічних налаштувань та підключення додаткових компонентів, щоб отримати певні функціональні можливості [23]. Це безпосередньо знову ж знижує поріг входу в процес розробки. Одним із плюсів так і одночасно з мінусів, є те, що Laravel містить велику кількість магічних реалізацій, що можуть як і спрощувати процес розробки для швидкості реалізації, так і ускладнювати, тому потребує додаткового розуміння в правильності використання певних речей.

Також в сторону спрощеності є відмінність у використовуваних ORM-бібліотек для роботи з базою даних [24]. В своєму контексті Symfony використовує сторонню бібліотеку Doctrine [25]. Вона слідує такому паттерну як

«Data mapper» [26]. У випадку Laravel використовується власна розробка Eloquent, яка реалізовує паттерн «Active record» [27]. В свою чергу, використання такої бібліотеки як Doctrine має сенс в більш масштабних проєктах з великою кількістю бізнес-логіки, в причину розділення сутностей від логіки для роботи з базою даних та повного дотримання принципів SOLID [28]. У випадку Eloquent кожна сутність являється відображенням себе в таблиці бази даних та самостійно відповідає за завантаження та зберігання себе до бази даних.

Враховуючи, що для розробки важлива спрощеність та низький поріг входу для подальшої підтримки іншими розробниками є доцільним обирання фреймворку Laravel.

Відповідно до бази даних в процесі проєктування було визначено, що найбільш актуальним для реалізації платформи є використання реляційних баз даних. Тому, опираючись на обрану серверну мову та фреймворк до неї абсолютно адекватно обрати таку базу даних як PostgreSQL [29], що є досить популярної та розповсюдженою у світи веб-розробки. Додатково, вона за замовчуванням підтримується Laravel і якісно справляється зі своїми задачами.

В якості середовища для розробки обрано спеціалізовану IDE PHPStorm з використанням індивідуальної ліцензії, оскільки вона практично являється монополістом з інструментів для розробки коду на PHP [30].

Для запуску серверної частини в локальній розробці буде використовуватись інструмент Docker, це досить популярне рішення в сьогоденні, щоб одночасно розробляти на специфічних версіях програмного забезпечення та позбутися його прямого встановлення на комп’ютер. Додатково це дає переваги при розробці в команді, оскільки у всіх розробників однакова конфігурація програмного забезпечення, що автоматично усуває можливі аномалії чи помилки в процесі роботи.

Для розробки клієнтської з елементами інтерактивності потрібно використовувати мову програмування JavaScript та мову розмітки HTML з додатковим використанням CSS для описання стилів. Вказаний стек технологій є стандартом для веб-додатків, тому це просто є необхідним.

Для зручності створення зручного та інтерактивного інтерфейсу, а також подальшої його підтримки доцільно обрати та використати один із frontend фреймворків. Найбільш актуальними та популярними на цей час рішеннями є Angular, React та Vue.js.

За своєю специфікою Angular має більший поріг входження та особливостей в плані створення архітектури додатку. При цьому він в певному обсязі поступає Vue.js та React в швидкості працездатності, а також націлений на розробку дійсно об`ємних проєктів, що прирівнюються до ентерпрайз рівня, де дійсно важлива архітектура з використанням відповідних патернів проєктування в користь перспективі довговічності проєкту.

Тепер доцільно порівняти безпосередньо React та Vue.js, оскільки вони в багатьом схожі між собою:

- використання Virtual DOM;
- надають реактивність та компонентну структуру;
- фокусуються на кореневій бібліотеці, виносячи інші питання, такі як роутинг або управління глобальним станом додатку в додаткові бібліотеки.

Відносно швидкості та оптимізації Vue.js має певні переваги. Наприклад, Vue на відміну від React вміє автоматично відстежувати залежності компоненту в момент його відтворення, в результаті цього система точно знає, які компоненти дійсно необхідно повторно відтворити в момент зміни його стану. В цілому та усуває необхідність у виконанні додаткових дій для оптимізації продуктивності додатку зі сторони розробника, а також дозволяє більше сфокусуватися на розробці.

Також, в представленні React всі елементи це JavaScript, включаючи структури HTML та CSS. В свою чергу Vue більше націлений на класичний контекст веб-технологій, тому ґрунтується на них. В контексті Vue існують так звані шаблони, що практично еквівалентні сучасному HTML, тому розробникам набагато зручніше працювати саме з ним. В результаті це зменшує поріг

входження та спрощує модифікацію додатку. Виходячи з описаної інформації доцільно обрати фреймворк Vue.js.

Отже, в контексті даного підрозділу було визначено та описано базові інструменти, що будуть використовуватись надалі при розробці програмної частини для реалізації інформаційної технології організації змагань з кібербезпеки.

3.2 Програмна реалізації системи

У відповідності з розробленої архітектурою та обраними інструментами були створені окремі проекти для серверної та клієнтських частин. Серверна частина є модульною, кожен з модулів був описаний на етапі створення архітекти програмної реалізації інформаційної технології організації змагань з кібербезпеки. Структура модулю Challenge наведена на рис. 3.1

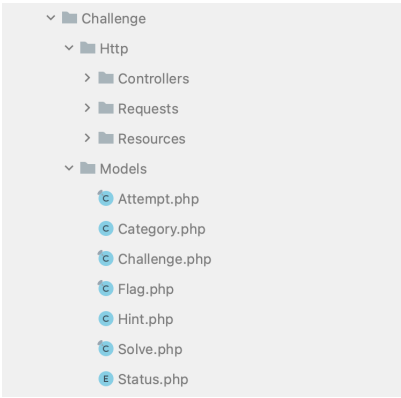


Рисунок 3.1 – Структура модулю Challenge

Опис структури модулю Challenge:

- Http – в даній директорії розміщується все, що відноситься до HTTP-запитів. Наприклад, контролери, що відповідають за обробку запитів та делегують виконання доменної логіки до сервісного слою. На рис. 3.2 зображено вміст контролеру, що займається обробку запиту на оновлення змагання.
- Models – містить собі в список сутностей предметної області в рамках модулю.
- Services – розміщення класів в яких реалізуються необхідна логіка в рамках предметної області.

```php
final class UpdateEventController
{
    public function __construct(
        private readonly Event $events,
        private readonly UpdateEventService $eventService,
    ) {
    }

    #[Route(methods: ['PUT', 'PATCH'], uri: 'events/:uuid', name: 'events.update')]
    public function __invoke(UpdateEventRequest $request, string $uuid): EventResource
    {
        $event = $this->eventService->update($this->events->findOrFail($uuid), $request->dto());

        return EventResource::make($event);
    }
}
```

Рисунок 3.2 – Вміст файлу UpdateEventController.php

В якості залежностей контролер приймає в конструкторі сутність Event та UpdateEventService. Об'єкт Event в даному випадку використовується для пошуку існуючого змагання в базі даних. UpdateEventService виконує логіку модифікації змагання та зберігання стану до бази даних. Список всіх модулів у проекті серверної частини зображено на рис. 3.3.

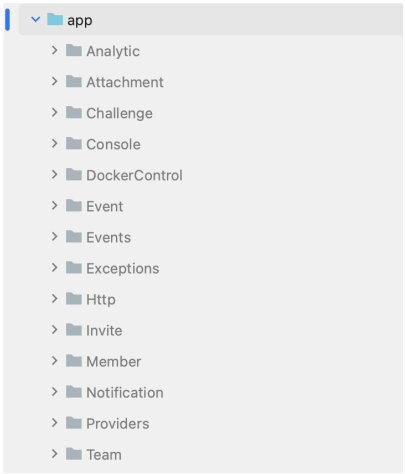


Рисунок 3.3 – Файлових лістинг всіх модулів серверної частини

Тепер розглянемо структуру клієнтської частини, що реалізована з використанням фреймворку Vue.js. Список компонентів клієнтської частини зображено на рис. 3.4.

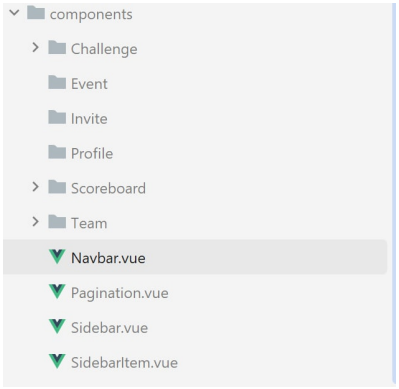


Рисунок 3.4 – Список компонентів клієнтської частини

В загальному структура та назви компонентів аналогічні до модулів серверної частини, що в даному випадку забезпечує єдність в термінах. Певні компоненти за необхідністю реалізують певну логіку та спілкуються з серверної частиною за допомогою спеціального API-клієнта. Файлова структура API-клієнта наведена на рис. 3.5.

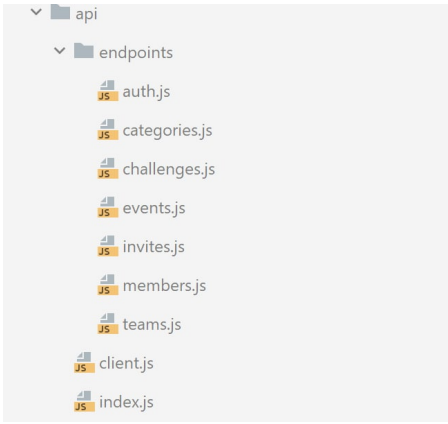


Рисунок 3.5 – Файлова структура API-клієнта

В рамках директорії endpoints описуються всі запити до серверної частини. Вміст файлу events.js зображено на рис. 3.6.

```
1  export default $client => ({
2    index() {
3      return $client.get('events');
4    },
5    show(eventId) {
6      return $client.get(`events/${eventId}`);
7    },
8    challenges(eventId) {
9      return $client.get(`events/${eventId}/challenges`);
10   },
11 });
```

Рисунок 3.6 – Вміст файлу events.js

Також, доцільно продемонструвати результати розробки клієнтської частини з відображенням у браузері. Вигляд сторінок змагання зображено на рис. 3.7 та 3.8.

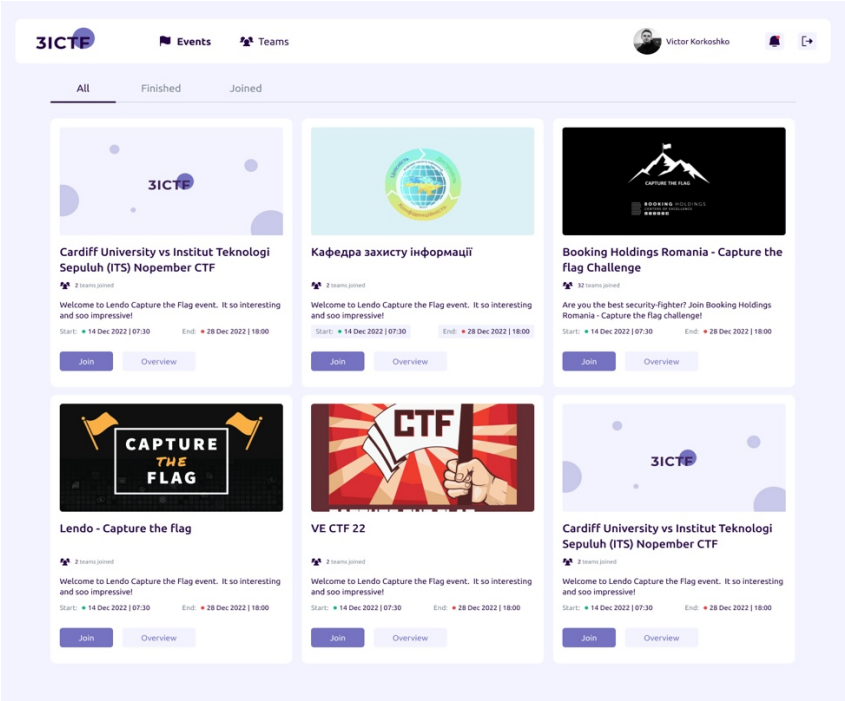


Рисунок 3.7 – Вигляд сторінки зі списком змагань в браузері

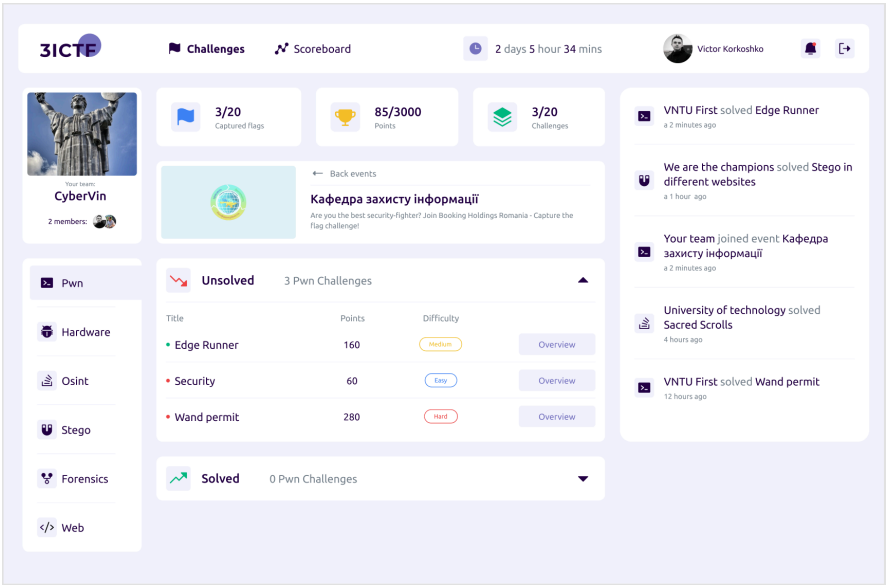


Рисунок 3.8 – Вигляд сторінки змагання в браузері

Отже, в рамках даного підрозділу було розроблено серверну та клієнтську частину, а також описано їх структуру в загальному вигляді. Також, наведено вигляд інтерфейсу клієнтської частини у браузері.

3.3 Програмна реалізації практичних завдань

Важливою частиною інформаційної технологіє є саме практичні завдання, тому в даному підрозділі буде наведена практична реалізація розроблених завдань у підрозділі 2.2.

2.2.1 Практична реалізації завдання типу «форензіка»

Відповідно до проведеного аналізу та формалізації завдання в категорії «комп'ютерна криміналістика» є необхідним вирішити наступні питання:

– формування дампу з трафіком в рамках тестової мережі;
– виконати емуляцію шкідливого запиту до одного з хостів;
– протестувати коректність сформованого дампу та наявність необхідного шкідливого запиту, котрий стане цілю для учасника змагання;
– протестувати можливість встановлення ідентифікатору CVE на базі шкідливої комбінації переданої в HTTP-запиті.

В якості інструменту для захоплення трафіку в мережі вирішено використовувати «Wireshark». Для початку встановимо його на тестову машину. Для цього необхідно перейти на офіційний сайт та завантажити інсталятор відповідно до використовуваної операційної системи. Блок з завантаженням інсталяторів на сайті Wireshark зображено на рис. 3.9.

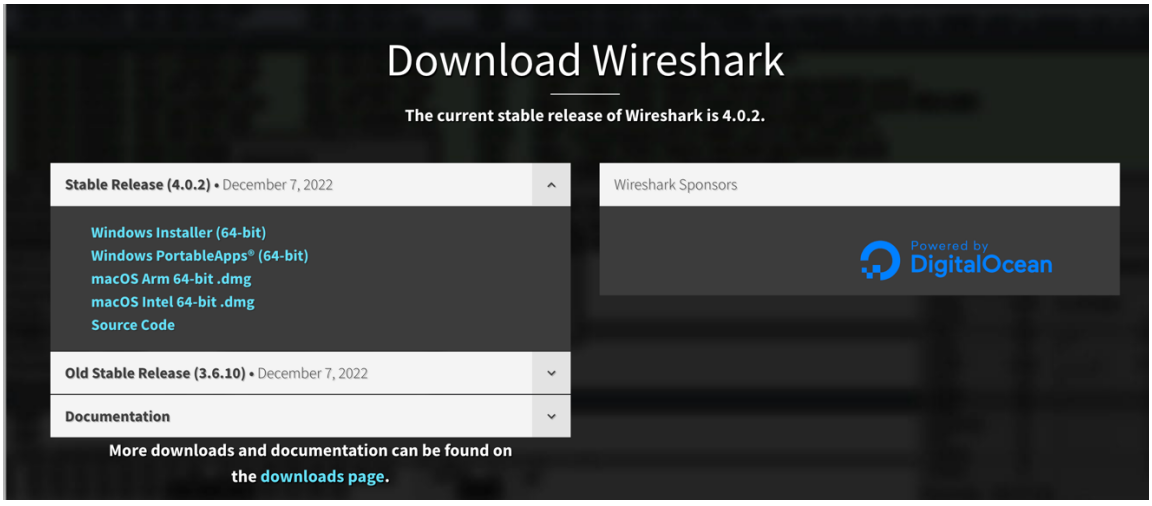


Рисунок 3.9 – Блок з завантаженням інсталяторів на сайті Wireshark

Як видно на рисунку 3.9, Wireshark пропонує інсталятори тільки для операційних систем Windows та macOS. На тестовій машині використовується операційна система macOS Monterey. Після завантаження та встановлення Wireshark може додатково знадобитись встановити додатковий драйвер для захвату трафіку на мережевих інтерфейсах.

Запускаємо встановлену програму та обираємо необхідний мережевий інтерфейс для захоплення трафіку. В поточному випадку буде використовуватись інтерфейс «en0». Вікно з вибором мережевого інтерфейсу в програмі Wireshark зображено на рис. 3.10.

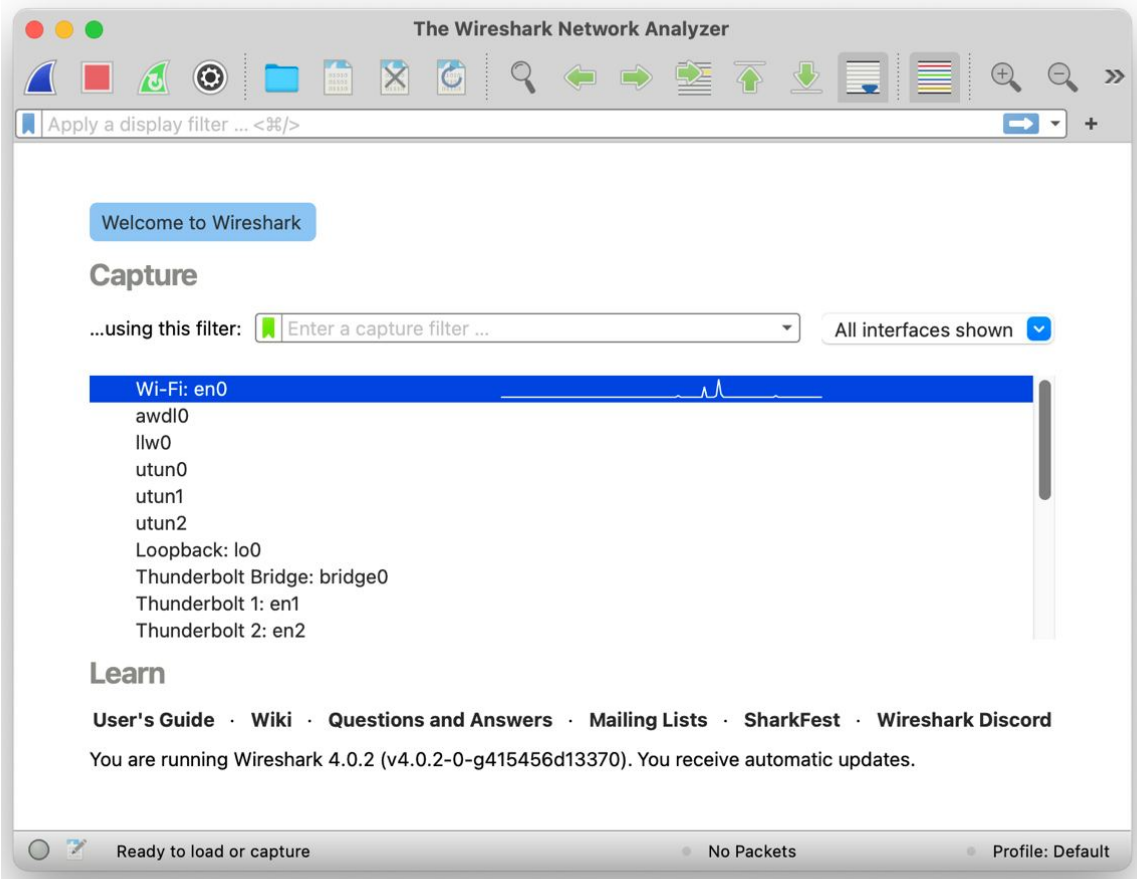


Рисунок 3.10 – Вікно з вибором мережевого інтерфейсу в програмі «Wireshark»

Для початку захоплення трафіку необхідно на панелі інструментів натиснути іконку з підказкою «Start capturing packets». Після чого інструмент почне процес захоплення пакетів на обраному мережевому інтерфейсі. Вигляд вікна з запущеним процесом захоплення зображено на рис. 3.11.

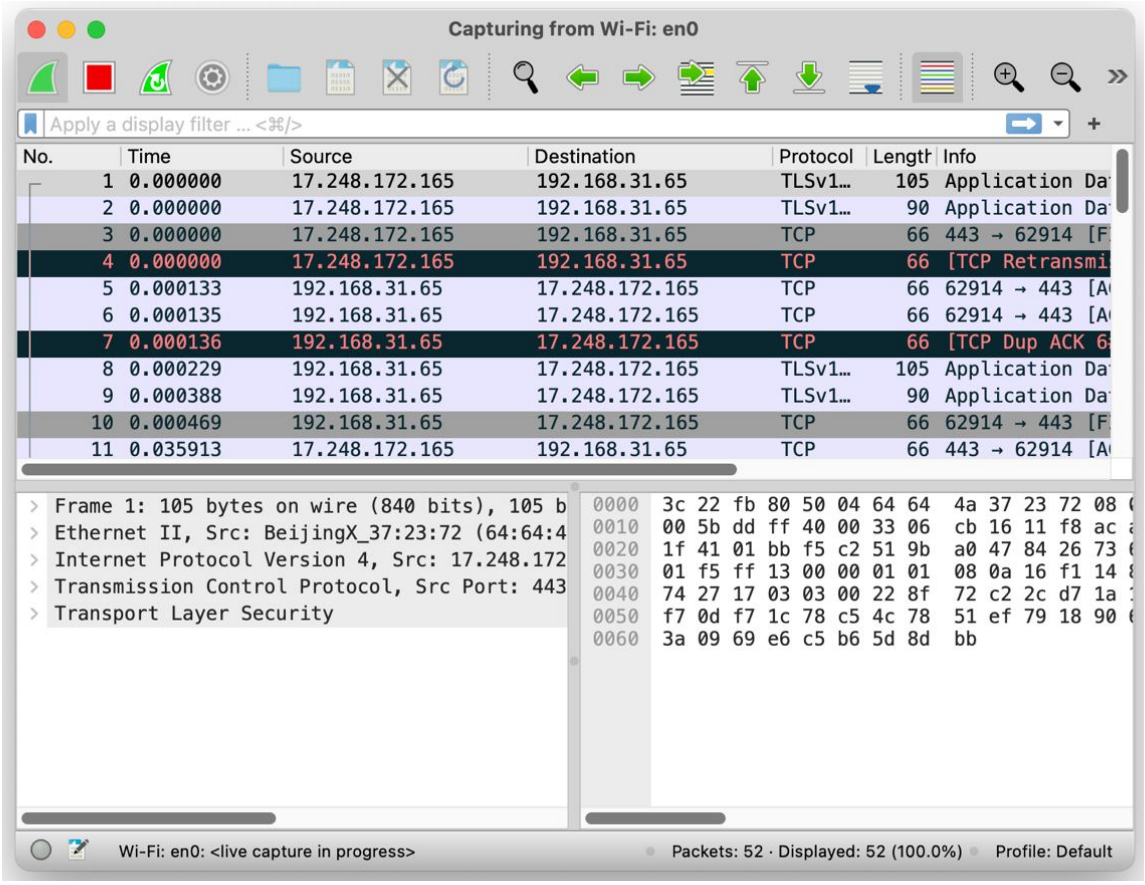


Рисунок 3.11 – Вигляд вікна з запущеним процесом захоплення пакетів

Процес захоплення пакетів відпрацьовує успішно. Тепер, необхідно створити емуляцію реального трафіку. Для цього пропонується запустити декілька HTTP-серверів на різноманітних портах. В рамках поточної задачі можна скористуватись влаштованим веб-сервером PHP. Приклад запуску веб-сервера наведено на рис. 3.12.

Рисунок 3.12 – Приклад запуску веб-сервера PHP

Як додаткове ускладнення для задачі може бути запуск декількох таких веб-серверів на різних портах та генерація великої кількості HTTP-запитів до них.

В поточному випадку виконаємо декілька тестових запитів та перевіримо їх наявність в Wireshark. Приклад тествого запиту зображено на рис. 3.13.

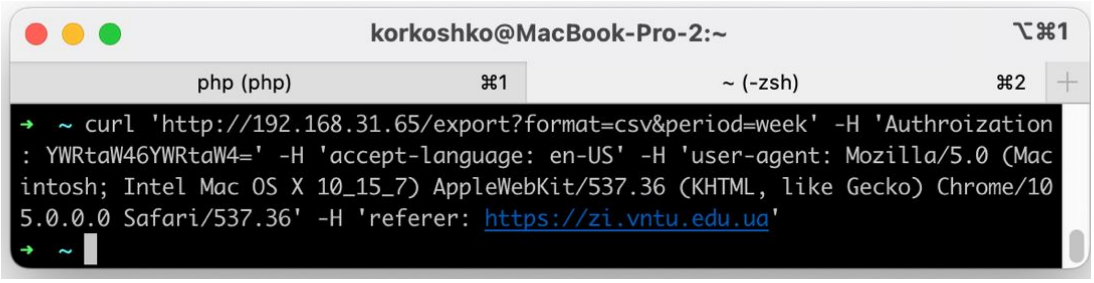


Рисунок 3.13 – Приклад тестового запиту до веб-серверу

В даному випадку відправляється GET запит до 192.168.31.65 з ціллю. Наявність відправленого запиту у Wireshark можна побачити на рис. 3.14.

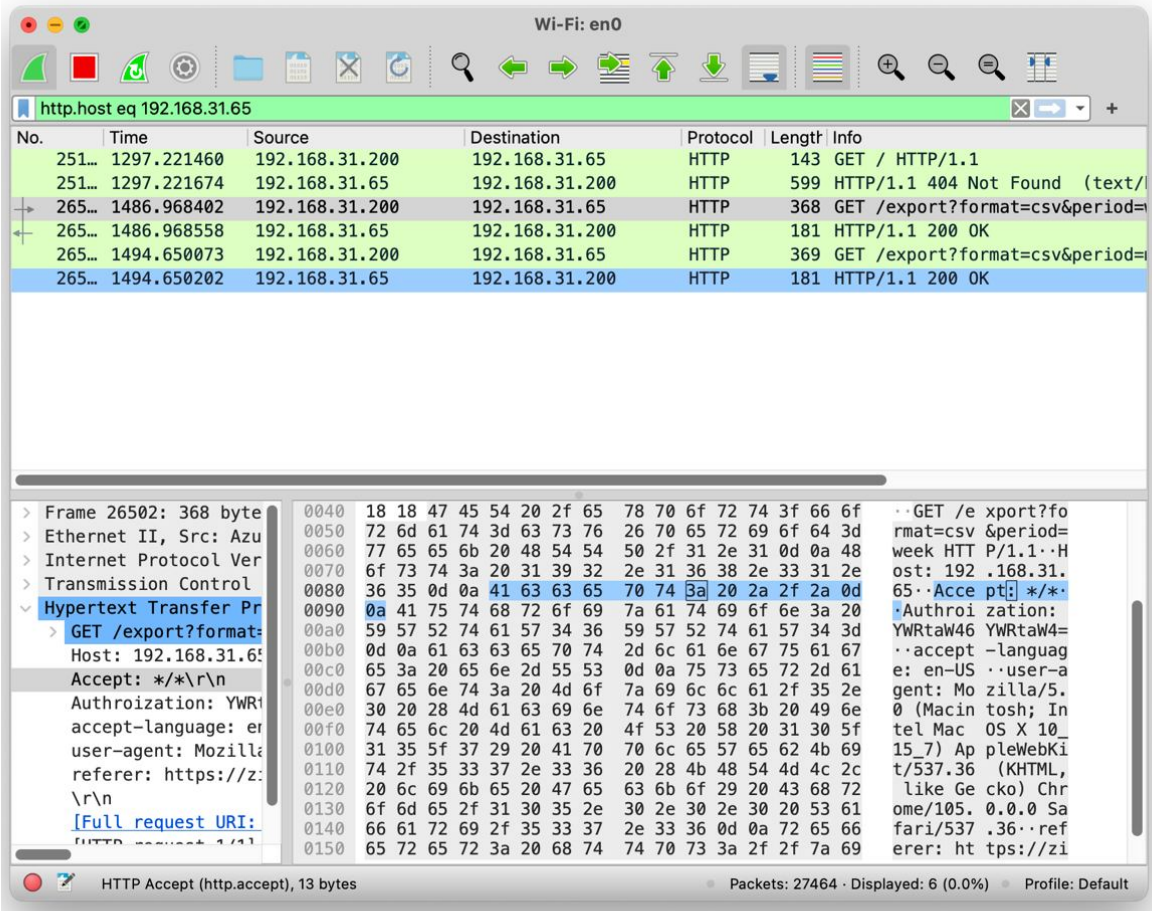


Рисунок 3.14 – Демонстрація захопленого тестового запиту у інструменті Wireshark

Наступним кроком є необхідним виконати шкідливий запит на базі CVE-2021-44228. Відправка шкідливого запиту до веб-серверу за адресою 192.168.31.65 зображено на рис. 3.15.

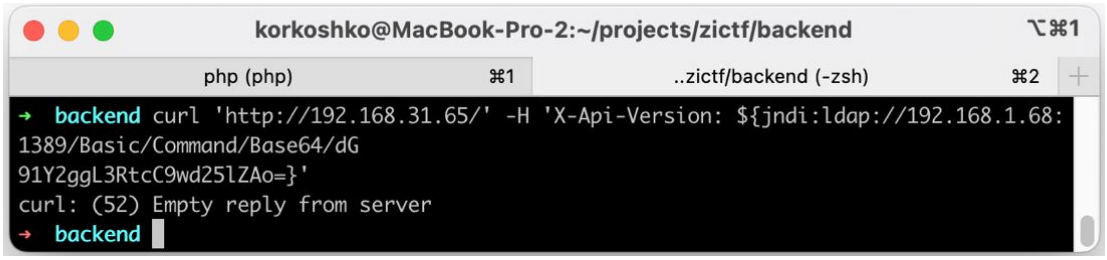


Рисунок 3.15 – Відправка шкідливого запиту до веб-серверу

Після чого виконуємо перевірку наявності шкідливого запиту у Wireshark. Результат перевірки зображено на рис. 3.16.

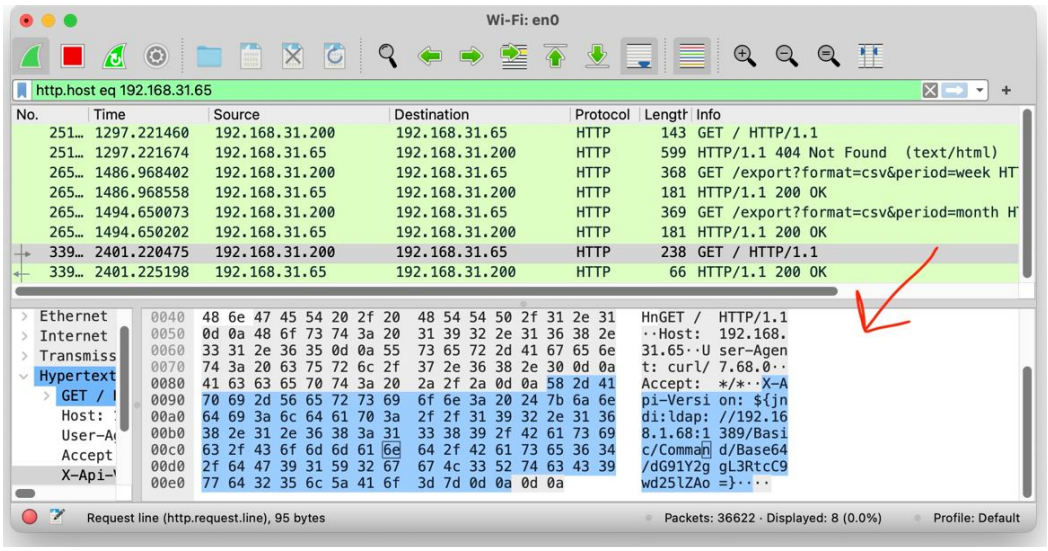


Рисунок 3.16 – Перевірка наявності шкідливого запиту у Wireshark

Таким чином практичний процес створення завдання завершено. Отриманий дамп необхідно завантажити в якості вкладеного файлу в процесі створення завдання за допомогою клієнтської частини організатора.

Покрокове виконання завдання відповідно сценарію від лиця учасника.

1) Ознайомлення з умовами завдання, що описані в рамках підрозділу 2.2.1.
2) Завантаження дампу трафіка, що містить сліди експлуатації вразливості.
3) Відкриття завантаженого дампу за допомогою інструменту Wireshark.
4) Пошук пакетів за протоколом HTTP використовуючи можливості фільтрації Wireshark.
5) Аналіз знайдених пакетів з попереднього кроку. В процесі аналізу виявлено, що значна кількість легітимних запитів містить в собі заголовок «Referer»,

тому є доцільним вилучити всі HTTP пакети, що містять в собі вказаний заголовок.

6) Для вилучення таких пакетів використано наступний фільтр: `http.request and not (http contains "Referer: ")`
7) В результаті отримано перелік з невеликою кількістю пакетів, які вже посильно опрацювати у ручному режимі. В одному із таких пакетів було знайдено підозріле значення у заголовку «X-Api-Version». Базуючись на ньому виконано пошук з ключовими словами «jndi» та «cve» за допомогою сервісу Google. Результат пошуку наведено на рис. 3.17.

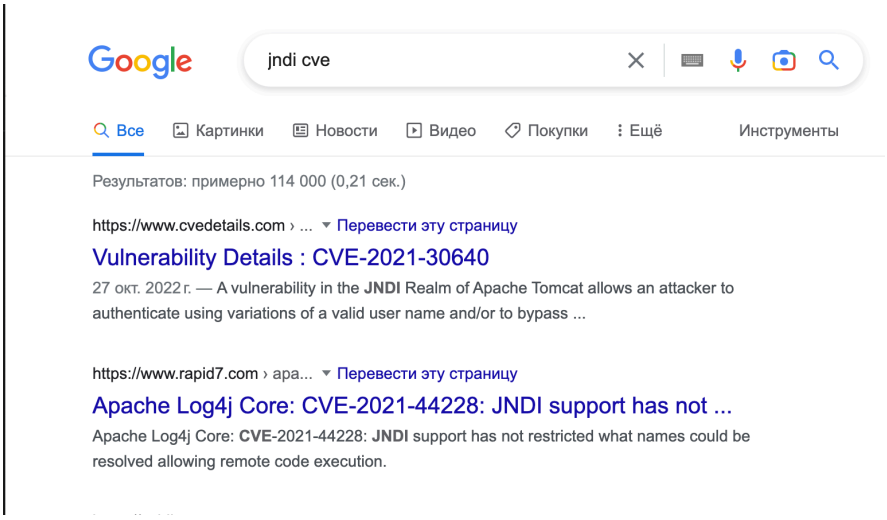


Рисунок 3.17 – Результат пошуку інформації про вразливість за ключовими словами

Після аналізу результатів пошуку вдалось зіставити знайдену вразливу комбінацію з тією, що описана в базі загальновідомих вразливостей з ідентифікатором CVE-2021-44228. Таким чином, завдання зі сторони учасника є виконаним. Прапорцем в даному випадку є безпосередньо ідентифікатор CVE.

3.2.2 Програмна реалізації завдання категорії «веб»

В даному підрозділі описується процес програмної реалізації завдання категорії «веб», що сформовано у підрозділі 2.2.2. Головною задачею відповідно до вимог є розробка веб-додатка вразливого до «GraphQL Batching Attack». Вразливий додаток розроблений з використанням Node.js та GraphQL. Також, він відповідає наступними вимогам:

– наявність головної сторінки сайту, що відповідає тематиці опису завдання;
– функція входу до кабінету за допомогою логіну та паролю.
– двофакторне підтвердження за допомогою одноразового паролю для входу до кабінету.
– відображення прапорця, після успішного входу до кабінету;
– Dockerfile з інструкціями встановлення необхідного програмного забезпеченням для роботи додатку.

Тому, відповідно до вимог було створено проект з структурую, що зображена на рис. 3.18.

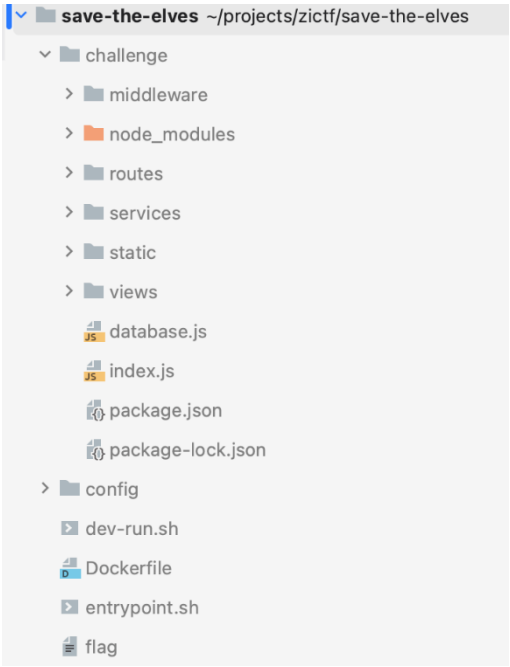


Рисунок 3.18 – Структура вразливого додатку категорії «веб».

В кореневій директорії проєкту розташовуються конфігураційні файли до використовуваного програмного забезпечення та безпосередньо текстовий файл з назвою «flag», що містить в собі прапорець, котрий відображається після виконання умови завдання. Вміст файлу з прапорцем зображено на рис. 3.19.

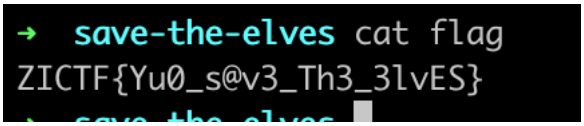


Рисунок 3.19 – Вміст файлу прапорця до завдання категорії «веб»

Вихідні файли самого вразливого додатку розташовуються в директорії «challenge». Створена схема API та мутації для входу до кабінету, а також для

перевірку двуфакторного підтвердження в контексті GraphQL зображено на рис. 3.20 та 3.21.

```
signIn: {
    type: ResponseType,
    args: {
        username: {type: new GraphQLNonNull(GraphQLString)},
        password: {type: new GraphQLNonNull(GraphQLString)}
    },
    resolve: async (root, args, {req, res}) => {
        return new Promise( executor: (resolve, reject) => {
            db.findUserForSignIn(args.username, args.password).then(async (user) => {
                if (!user.length) reject(new Error('Invalid credentials!'));

                const token = await jwtService.sign( data: {username: user[0].username, verified: false});

                res.cookie('session', token, {maxAge: 3600000});
                resolve( value: {message: "User logged in successfully!", token: token});

            }).catch(err => reject(new GraphQLError(err)));
        });
    }
},
```

Рисунок 3.20 – Реалізація мутації входу користувача за допомогою GraphQL

В даному випадку мутація приймає два аргументи, а саме «username» та «password», після чого шукає користувача в базі даних, що відповідає значеннями переданих аргументів, якщо такий користувач існує, то створюється JWT токен та відправляється в якості cookies клієнту. В іншому випадку відправляється помилка, що надані дані для входу не є дійсними.

```
verify2FA: {
    type: ResponseType,
    args: {
        otp: {type: new GraphQLNonNull(GraphQLString)}
    },
    resolve: async (root, args, {req, res}) => {
        return new Promise( executor: async (resolve, reject) => {
            if (!req.user) return reject(new GraphQLError('Authentication required!'));

            const user = await db.findUserForSignIn(req.user.username);
            if (await otpService.verify(user.otp_secret, args.otp)) {
                let token = await jwtService.sign( data: {username: req.user.username, verified: true});
                res.cookie('session', token, {maxAge: 3600000});
                resolve( value: {message: "2FA verified successfully!", token: token});
            } else reject(new GraphQLError(new Error('Invalid OTP!')));
        });
    }
}
```

Рисунок 3.21 – Реалізація мутації двофакторного підтвердження за допомогою GraphQL

Мутація двофакторного підтвердження приймає на вхід аргумент з іменем «otp». Після чого перевіряє наявність JWT токена у користувача, далі з бази даних витягується спеціальний секрет, на базі цього секрету виконується верифікація відправленого одноразового паролю. У випадку успіху користувачу генерується новий JWT токен з міткою, що він є верифікованим та по аналогії за допомогою cookies відправляється на клієнт. Саме з цією мутацією необхідно взаємодіяти учаснику змагання та обійти двуфакторне підтвердження з використанням пакетних запитів GraphQL.

Також, для закриття доступу до кабінету реалізовано спеціальний middleware в якому міститься логіка аутентифікації та авторизації користувача, вміст якого зображено на рис. 3.22.

```
module.exports = async (req, res, next) => {
    try {
        const jwtToken = req.cookies.session;

        if (jwtToken === undefined) {
            if (req.originalUrl === '/graphql') return next();
            if (!req.is('application/json')) return res.redirect('/');

            return res.status(401).json({status: 'unauthorized', message: 'Authentication required!'});
        }

        return jwtService.verify(jwtToken).then(user => {
                req.user = user;

                if (req.originalUrl === '/graphql' || req.originalUrl === '/2fa') return next();
                if (user.verified === false) return res.redirect('/login');

                return next();
            })
            .catch((e) => {
                if (req.originalUrl === '/graphql') return next();

                res.redirect('/logout');
            });
    } catch (e) {
        console.log(e);
        return res.redirect('/logout');
    }
}
```

Рисунок 3.22 – Вміст файлу AuthMiddleware.js

Аутентифікація в даному випадку виконується за допомогою JWT токена, що передається в якості cookies. В контексті поточного додатку процес авторизації досить простий, якщо користувача автентифіковано – він має доступ до перегляду сторінки кабінету на якій відображається прапорець.

Для виключення сценарію, коли учасник змагання може схитрувати та розпочати процес звичайної атаки методом грубого перебору передбачено налаштування обмеження швидкості за допомогою веб-серверу NGINX [31]. Конфігурація веб-серверу NGINX для вразливого додатку категорії «веб» зображена на рис. 3.23.

```
limit_req_zone global zone=global_rate_limit:5m rate=20r/m;
limit_req_status 429;

server {
    listen 80;
    server_name save-the-elves.zictf;

    location /graphql {
        limit_req zone=global_rate_limit burst=5 nodelay;
        proxy_set_header  X-Forwarded-For $remote_addr;
        proxy_set_header  Host $http_host;
        proxy_pass    http://127.0.0.1:1337;
    }

    location / {
        proxy_set_header  X-Forwarded-For $remote_addr;
        proxy_set_header  Host $http_host;
        proxy_pass        http://127.0.0.1:1337;
    }
}
```

Рисунок 3.23 – Конфігурація веб-серверу NGINX для вразливого додатку категорії «веб»

В даному випадку для віртуального хоста «save-the-elves.zictf» на глобальному рівні налаштовано 20 запитів в секунду, а також додатково для запитів за шляхом /graphql вказана опція burst зі значенням 5, що відповідає за кількість одночасних запитів в черзі, якщо в черзі одночасно знаходиться більше ніж 5 запитів, то клієнту повертається помилка з кодом відповіді 503 Service

Unavailable. Таким чином, звичайна атака методом грубого перебору не є можливою в причину малої ймовірності співпадіння, оскільки одноразовий пароль дійсний тільки в рамках 30 секунд та забезпечує унікальність на базі часу.

Також, сформовано Dockerfile, що містить в собі набір інструкцій для створення зображення Docker на базі якого виконується запуск контейнера системою інформаційної технології організації змагань з кібербезпеки. Вміст файлу Dockerfile зображено на рис. 3.24.

```
FROM node:18-alpine3.15

# Install system packages
RUN apk add --update --no-cache supervisor mariadb mariadb-client gcc musl-dev \
    curl nginx

# Setup application
RUN mkdir -p /opt/exports && chown node:node /opt/exports
RUN mkdir -p /app

# Copy flag
COPY flag /flag

# Add application
WORKDIR /app
COPY challenge .

# Install dependencies
RUN npm install --legacy-peer-deps

# Setup superivsord
COPY config/supervisord.conf /etc/supervisord.conf
COPY config/nginx.conf /etc/nginx/nginx.conf

# Expose the port node-js is reachable on
EXPOSE 80

# Seed database and start supervisord
COPY --chown=root entrypoint.sh /entrypoint.sh
RUN chmod +x /entrypoint.sh
ENTRYPOINT /entrypoint.sh
```

Рисунок 3.24 – Вміст Dockerfile для вразливого додатку категорії «веб»

В зазначеному Dockerfile в якості базового зображення використовується «node:18-apline3.15», який вже містить в собі встановлений Node.JS версії 18, що необхідний для роботи додатку. Наступними операціями виконуються команди

встановлення додаткових пакетів програмного забезпечення у вигляді СКБД MariaDB та веб-серверу NGINX. Далі виконуються інструкції створення директорії в який копіюються файли додатку, що містяться в директорії «challenge», а також файл з вмістом прапорця. Після чого виконується встановлення необхідних залежностей вразливого додатку за допомогою пакетного менеджера NPM [32]. Далі копіюються конфігураційні файли для веб-серверу та в якості вхідної точки встановлюється shell-скрипт, що буде виконуватись при старті контейнеру. В скрипті entrypoint.sh містяться інструкції, що створюють базу даних та заповнюються її тестовими даними, а також запускають служби необхідних сервісів за допомогою інструменту Supervisor [33].

Таким чином, процес програмної реалізації вразливого веб-додатку завершено. Вихідні файли з додатком та конфігурацією необхідно запакувати в архів формату ZIP [34]. Після чого при додаванні завдання до системи інформаційної технології за допомогою клієнтської частини організатора завантажити його в якості файлу з поміткою, що це програмна реалізація завдання. За допомогою цього система автоматично проаналізує вміст архіву та зможе автоматично розгортати додаток за допомогою Docker під кожну з команду завдяки описаному Dockerfile [35].

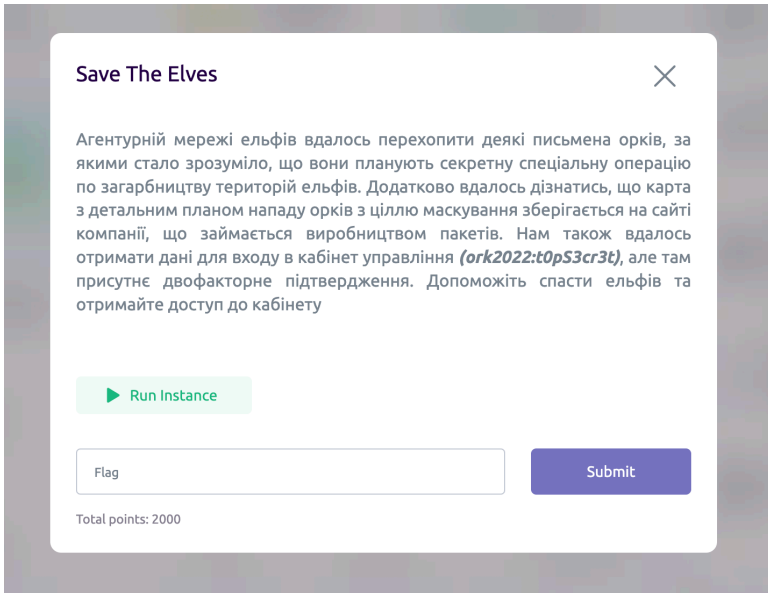


Рисунок 3.25 – Вигляд завдання на рівні клієнтської частина учасника

Покрокове виконання завдання відповідно до сценарію від лиця учасника.

1) Ознайомлення з умовами завдання, що описані в рамках підрозділу 2.2.2. Додатково вигляд поточного завдання в рамках клієнтської частини учасника зображено на рис. 3.26
2) Запуск контейнеру завдання за допомогою кнопки «Run instance».
3) Перехід за посиланням на сторінку вразливого додатку.
4) Спроба увійти до кабінету за допомогою відомих облікових даних користувача, що надані в рамках умови завдання.
5) В результаті входу отримано інформацію про довжину коду підтвердження (4 цифри).
6) Дослідження відправлених запитів до сервера за допомогою консолі розробника в браузері. В результаті чого з'ясовано, що запити для входу та підтвердження відправляють до GraphQL API.
7) Спроба передачі різних текстових комбіній в якості коду підтвердження. В результаті чого ніяких успіхів не досягнено.
8) Спроба виконати атаку грубого підбору, що також без успіхів по причині наявності обмежувача запитів зі сторони веб-серверу.
9) Пошук додаткової інформації про можливі атаки в GraphQL та обхід двуфакторної автентифікації. В результаті чого знайдено інформацію про «GraphQL Batch Attack».
10) Дослідження знайденої атаки. Написання shell-скрипта для відправки пакетних запитів для підбору коду підтвердження, що дозвовило обійти обмеження запитів зі сторони веб-сервера та успішно аутентифікуватись.
11) В результаті аутентифікація отримано JWT токен. Встановлюємо в браузері отриман токен в якості значення сесії в кукі, після чого успішно отримуємо доступо до кабінету, де відображається прапорець.

3.4 Аналіз результатів змагання

Для тестування програмної реалізації системи в рамках інформаційної технології організації змагань з кібербезпеки було проведено змагання на базі

розроблених завдань в попередній розділах. Для виконання цієї дії було створено змагання за допомогою клієнтської частини організатора. Вигляд сторінки створення змагання зображено на рис. 3.26.

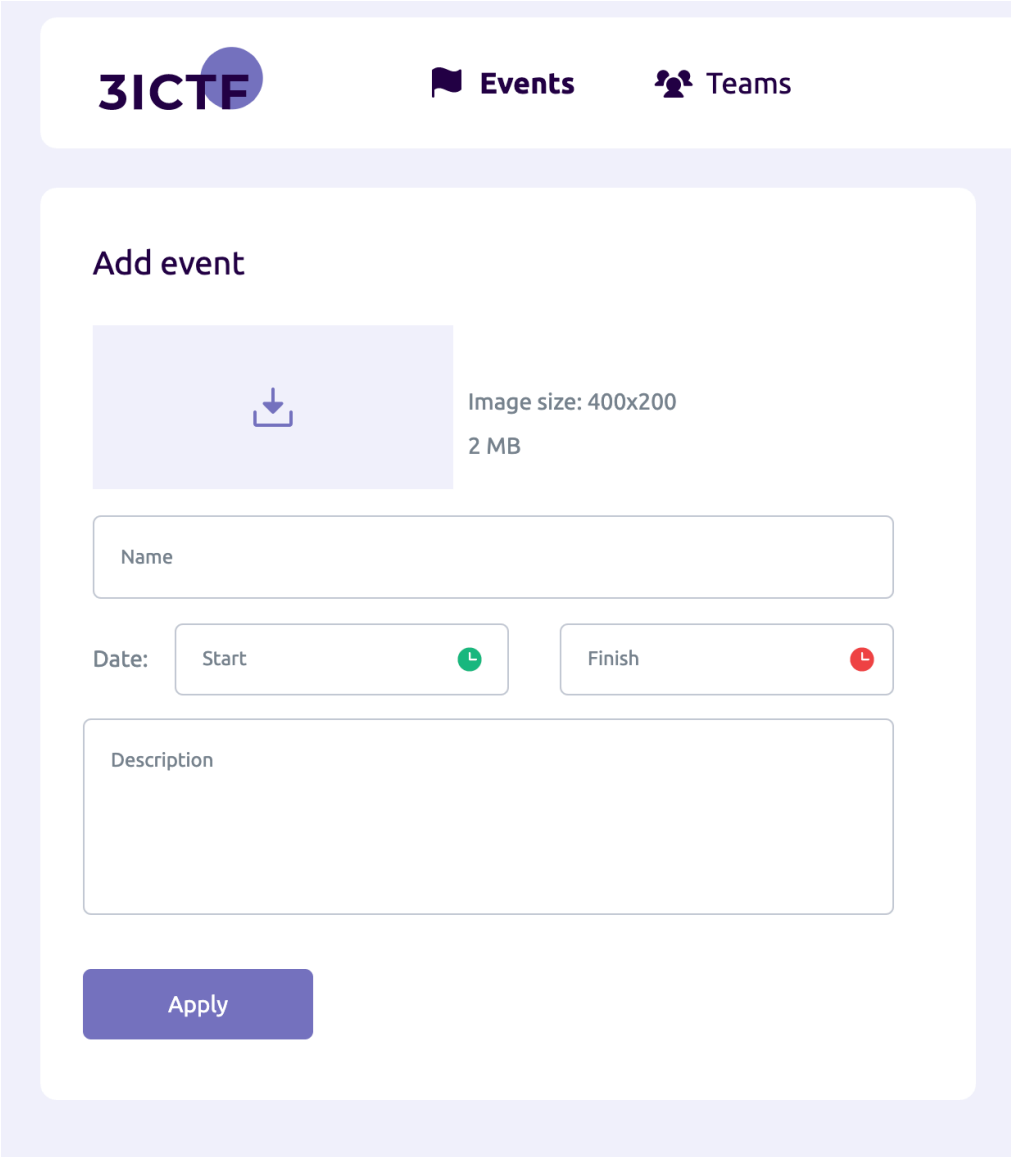


Рисунок 3.26 – Вигляд сторінка створення завдання на рівні клієнтської частини організатора

Наступником кроком для створеного змагання було додано безпосередньо самі завдання, що були описані та розроблені в рамках підрозділів 2.2 та 3.2. Вигляд сторінки створення завдання зображено на рис. 3.28.

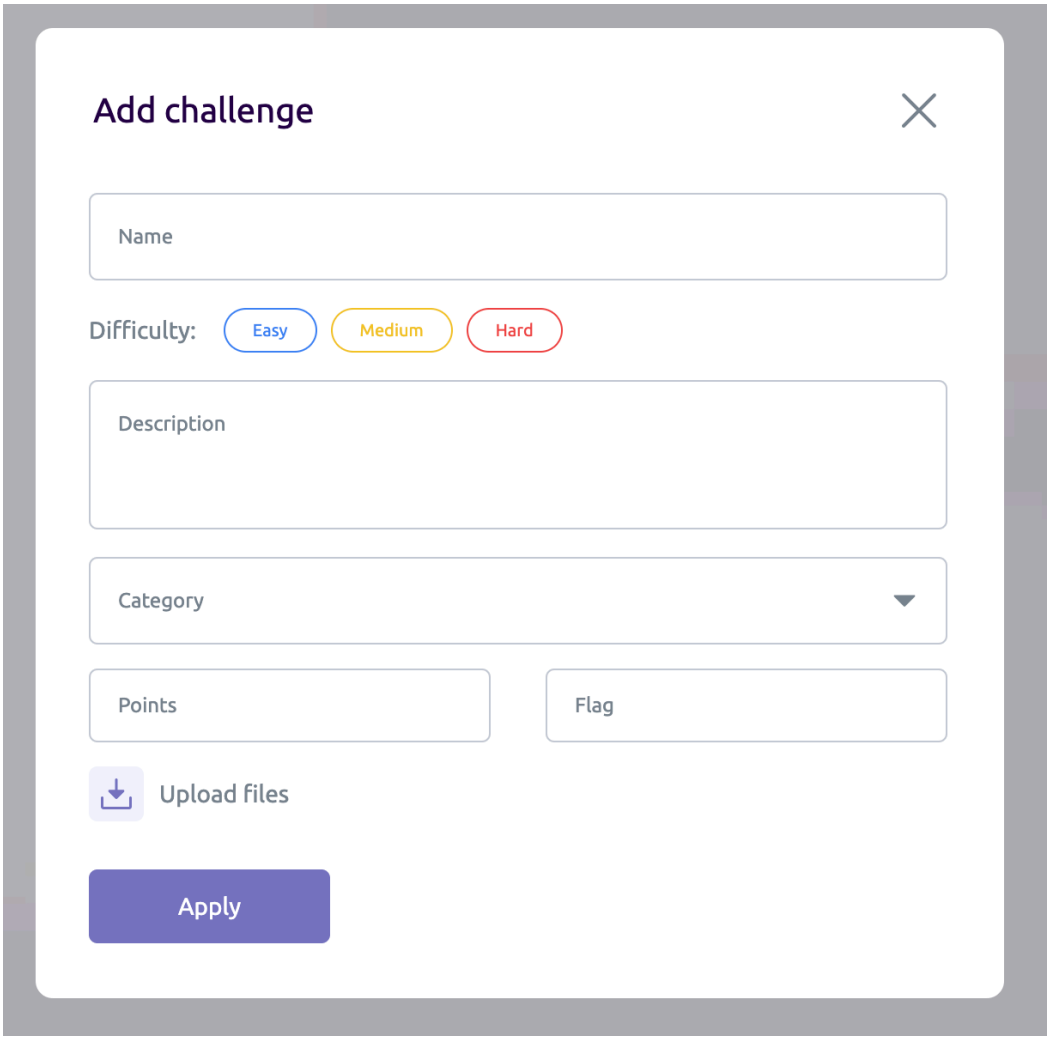


Рисунок 3.27 – Вигляд сторінки створення завдання на рівні клієнтської частина організатора

Після завершення підготовки на рівні системи необхідно було запросити учасників для участі у змаганні. Для різноманітності результатів та відображення більш реалістичної картини відносно необхідного фаху для виконання розроблених завдань було запрошено 2-х фахівців з кібербезпеки та 2-х IT-фахівців, що не мають фаху в рамках безпеки комунікаційних та інформаційних систем. Список всіх учасників наведено в таблиці 3.1.

Ім’я учасника	Позиція
Хилько Степан	Фахівець з кібербезпеки
Борусевич Артур	Фахівець з кібербезпеки
Кабачій Богдан	Тестувальник
Ільницький Ярослав	Розробник

Таблиця 3.1 – Список учасників проведеного змагання

Наступним етапом був процес проведення самого змагання. Учасникам змагання було виділено на вирішення завдань 48 годин, що дало їм змогу взяти участь у зручний їм час враховуючі поточні обставини сьогодення з хаотичним виключенням світла.

В рамках процесу проведення змагання були виявлені певні проблеми з автоматичним запуском контейнеру, що містить в собі вразливий додаток в категорії «веб». Проблеми швидко були усунуті завдяки комунікації з учасниками за допомогою спільного чату в месенджері Telegram. По закінченню 48 годин, змагання в автоматичному режимі було завершено та закрито доступ до завдань з можливістю надсилати прапорці для перевірки.

Подальшим кроком після завершення змагання було виконання аналізу результатів, які досягли учасники. Результати наведені в таблиці 3.2.

Завдання	Учасник	Виконав	Час виконання
Форензіка	Хилько Степан	+	34 хв.
	Борусевич Артур	+	22 хв.
	Кабачій Богдан	+	4 год. 45 хв.
	Ільницький Ярослалв	+	3 год. 12 хв.
Веб	Хилько Степан	+	3 год. 22 хв.
	Борусевич Артур	+	5 год. 4 хв.
	Кабачій Богдан	-	-
	Ільницький Ярослалв	-	-

Таблиця 3.2 – Результати проведеного змагання

Відповідно до результатів можна зробити висновок, що проведене змагання пройшло досить успішно. Учасникам вдалось виконувати завдання.

Також, грунтуючись на кількості загальних рішень в контексті завдання , можна визначити, що найбільш складним завданням було завдання з категорії «веб». На рис. 3.28 зображена діаграма по загальній кількості вирішених завдань.

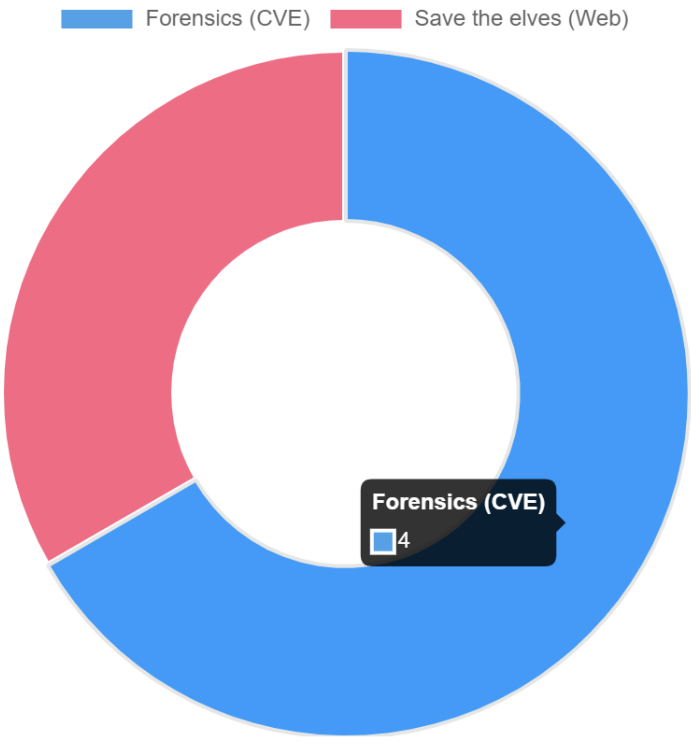


Рисунок 3.27 – Діграма загальної кількості рішень по завданням

В свою чергу найбільш легким виявилось завдання з категорії «форензіка», але кожен з учасникі вирішив це завдання за різний час. Графік затраченого часу на виконання кожного з завдань учасниками наведено на рис. 3.28.

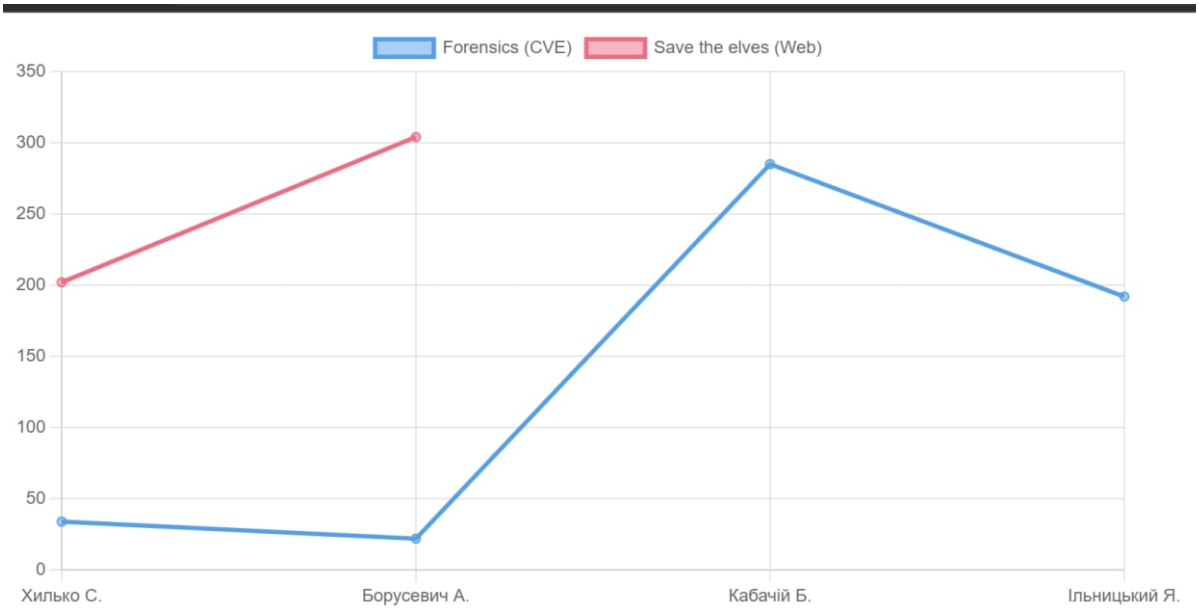


Рисунок 3.28 – Графік затраченого часу в хвилиних на виконання кожного завдання

Результатами на рис. 3.28 можна підтвердити зроблені висновки відносно складності завдань. Але, час виконання заміряється, наприклад, від запуску контейнеру з додатком до моменту, коли учасник відправить коректний прапорець для підтвердження виконання завдання. Тому, вказаний параметр в певних випадках не може гарантувати реальний час виконання завдання учасником, але відображає загальну тенденцію.

Заключним етапом ознайомось з графіком найкращих учасників, що зображено 3.29.

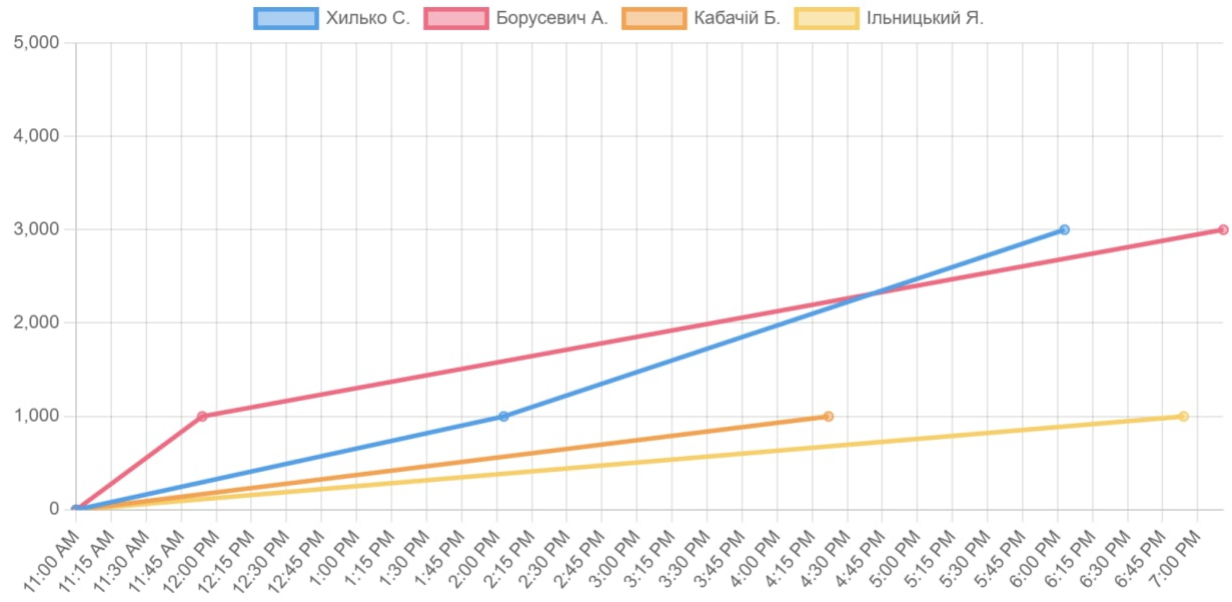


Рисунок 3.29 – Графік найкращих учасниів

Відповідно до графіку на рис. 3.29 можна побачити кількість балів, що отримали учасники за виконання завдання та виявити таким чином переможців.

Отже, можна зробити висновок, що найбільш ефективно та швидко виконали всі завдання тільки учасники, що мають необхідних фах в області безпеки інформаційних та комунікаційних систем. Два інших учасника, що являються фахівцями в IT-сфері змогли виконати тільки завдання з категорії «форензіка» та при цьому уступаючи в часі виконання фахівцям з кібербезпеки. Цей факт дає зрозуміти, що завдання виконані на якісному рівні та відповідно фахових навичок спеціалістів з кібербезпеки.

4 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження «Інформаційна технологія організації змагань з кібербезпеки» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

4.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія організації змагань з кібербезпеки» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.1 [35].

Таблиця 4.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					

1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	5	5
2. Ринкові переваги (наявність аналогів)	2	3	2
3. Ринкові переваги (ціна продукту)	2	2	3
4. Ринкові переваги (технічні властивості)	1	2	2
5. Ринкові переваги (експлуатаційні витрати)	2	2	2
6. Ринкові перспективи (розмір ринку)	3	4	3
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	5	5	5
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	4	5	5
11. Практична здійсненність (термін реалізації)	4	4	4
12. Практична здійсненність (розробка документів)	4	3	4
Сума балів	37	41	41
Середньоарифметична сума балів $СБ_c$	39,7		

Таблиця 4.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами.

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки зведено до таблиці 4.2.

За результатами розрахунків, наведених в таблиці 4.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.3 [35].

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Таблиця 4.3 – Науково-технічні рівні та комерційні потенціали розробки

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія організації змагань з кібербезпеки» становить 39,7 бала, що, відповідно до таблиці 4.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

4.2 **Розрахунок узагальненого коефіцієнта якості розробки**

Окрім комерційного аудиту розробки доцільно також розглянути технічний рівень якості розробки, розглянувши її основні технічні показники. Ці показники по-різному впливають на загальну якість проектної розробки.

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення розрахуємо за формулою [36]:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i , \qquad (4.1)$$

де k – кількість найбільш важливих технічних показників, які впливають на якість нового технічного рішення;

α_i – коефіцієнт, який враховує питому вагу *і-го* технічного показника в загальній якості розробки.

Коефіцієнт α_i визначається експертним шляхом і при цьому має виконуватись умова $\sum_{i=1}^{k} \alpha_i = 1$;

β_i – відносне значення *і-го* технічного показника якості нової розробки.

Відносні значення β_i для різних випадків розраховуємо за такими формулами:

для показників, зростання яких вказує на підвищення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ні}}{I_{ai}}, \tag{4.2}$$

де $I_{ні}$ та $I_{на}$ – чисельні значення конкретного *і*-го технічного показника якості відповідно для нової розробки та аналога;

для показників, зростання яких вказує на погіршення в лінійній залежності якості нової розробки:

$$\beta_i = \frac{I_{ai}}{I_{ні}}; \tag{4.3}$$

Використовуючи наведені залежності можемо проаналізувати та порівняти техніко-економічні характеристики аналогу та розробки на основі отриманих наявних та проектних показників, а результати порівняння зведемо до таблиці 4.4.

Таблиця 4.4 – Порівняння основних параметрів розробки та аналога.

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
універсальність	бал	7	8,5	1,21	0,25
надійність	%	80	95	1,19	0,15
маштабованість	%	50	75	1,5	0,1
кількість змагань одночасно	шт.	1	10	10	0,3
аналітика по змаганням	шт.	4	12	3	0,2

Узагальнений коефіцієнт якості ($B_н$) для нового технічного рішення складе:

$$B_н = \sum_{i=1}^{k} \alpha_i \cdot \beta_i = 1{,}21 \cdot 0{,}25 + 1{,}19 \cdot 0{,}15 + 1{,}5 \cdot 0{,}1 + 10 \cdot 0{,}3 + 3 \cdot 0{,}2 = 4{,}23.$$

Отже за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 4,23 рази.

4.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов'язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія організації змагань з кібербезпеки», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

4.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [35].

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (4.4)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=24 дні.

$$З_о = 17200{,}00 \cdot 24 / 24 = 17200{,}00 \text{ грн.}$$

Проведені розрахунки зведемо до таблиці.

Таблиця 4.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	17200,00	716,67	24	17200,00
Консультант (фахівець з організації змагань, олімпіад тощо)	17050,00	710,42	10	7104,17
Інженер-програміст 1-ї категорії	17000,00	708,33	24	17000,00
Технік	7300,00	304,17	20	6083,33
Всього				47387,50

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія організації змагань з кібербезпеки» розраховуємо за формулою:

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i, \quad (4.5)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}}, \quad (4.6)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [35];

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих

об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 24 дн;

$t_{зм}$ – тривалість зміни, год.

$C_1 = 6700{,}00 \cdot 1{,}35 \cdot 1{,}65 / (24 \cdot 8) = 77{,}73$ *грн.*
$З_{р1} = 77{,}73 \cdot 11{,}00 = 855{,}04$ *грн.*

Таблиця 4.8 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Встановлення серверного обладнання для проведення досліджень	11,00	3	1,35	77,73	855,04
Інсталяція програмного забезпечення розробки програмного забезпечення	7,05	3	1,35	77,73	548,00
Встановлення цифрових обчислювальних систем	6,00	5	1,70	97,88	587,30
Відлагодження програмних модулів формування завдань	5,75	5	1,70	97,88	562,83
Підготовка тестового дослідження	9,12	4	1,50	86,37	787,67
Формування бази даних результатів випробування системи	22,00	3	1,35	77,73	1710,07
Всього					5050,90

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (4.7)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод} = (47387{,}50 + 5050{,}90) \cdot 10 / 100\% = 5243{,}84$ *грн.*

4.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (4.8)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (47387,50 + 5050,90 + 5243,84) · 22 / 100% = 12690,09 грн.

4.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія організації змагань з кібербезпеки».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$М = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{вj}, \quad (4.9)$$

де $Н_j$ – норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ = *1,1 … 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 3,0 · 225,00 · 1,1 - 0 · 0 = 742,50 грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.9 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір канцелярський офісний (А4)	225,00	3,0	0	0	742,50
Папір для заміток (А5)	116,00	3,0	0	0	382,80
Начиння канцелярське	195,00	3,0	0	0	643,50

Органайзер офісний	183,00	3,0	0	0	603,90
Картридж для принтера	950,00	2,0	0	0	2090,00
Всього					4462,70

4.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія організації змагань з кібербезпеки», розраховуємо, згідно з їхньою номенклатурою, за формулою:

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (4.10)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j = 1,1 \ldots 1,15$).

$К_в = 1 \cdot 1560,00 \cdot 1,1 = 1716,00$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.10 – Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
Зовнішній жорсткий диск 2.5" 1TB Seagate (STGD2000200)	1	1560,00	1716,00
Концентратор Defender SEPTIMA SLIM (83505)	1	400,00	440,00
Кабель для передачі даних USB to COM 1.0m Patron (CAB-PN-USB-COM)	1	354,00	389,40
Всього			2545,40

4.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного

для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (4.11)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 37846,00 \cdot 1 \cdot 1,1 = 41630,60$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.11 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Серверне обладнання на основі: процесор: Intel Xeon E5-2630 v4 та подібні до них; - оперативна пам'ять: від 4Gb; - операційна система: Linux - дисковий простір: від 25 GB - тип накопичувача: SSD	1	37846,00	41630,60
Всього			41630,60

4.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних

для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (4.12)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 50,00 \cdot 1 \cdot 1,1 = 55,00$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 4.12 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Середовище розробки: PHPStorm з ліцензією for individual use	1	50,00	55,00
Всього			55,00

4.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12} \ , \qquad (4.13)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (42975{,}00 \cdot 1) / (2 \cdot 12) = 1790{,}63$ *грн.*

Проведені розрахунки зведемо до таблиці.

Таблиця 4.13 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Macbook 2019 Pro 16 Характеристики: - процесор: Intel Core i7 - оперативна пам'ять: 16GB (DDR4) - операційна система: macOS Monterey	42975,00	2	1	1790,63
Місце оператора спеціалізоване	9400,00	5	1	156,67
Офісна оргтехніка	9650,00	4	1	201,04
Лабораторія досліджень програмного забезпечення	520000,00	25	1	1733,33
Всього				3881,67

4.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (4.14)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_е = 6,20$ грн;

$К_{вп і}$ – коефіцієнт, що враховує використання потужності, $К_{вп і} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$В_е = 0,05 \cdot 240,0 \cdot 6,20 \cdot 0,95 / 0,97 = 74,40$ грн.

Проведені розрахунки зведемо до таблиці.

Таблиця 4.14 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Macbook 2019 Pro 16 Характеристики: - процесор: Intel Core i7 - оперативна пам'ять: 16GB (DDR4) - операційна система: macOS Monterey	0,05	240,0	74,40
Програмно-обчислювальний комплекс розробки програмного забезпечення	0,42	240,0	624,96
Місце оператора спеціалізоване	0,10	240,0	148,80
Офісна оргтехніка	0,60	6,0	22,32
Всього			870,48

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія організації змагань з кібербезпеки» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» відсутні.

4.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» відсутні.

4.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.15)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів} = 50\%$.

$I_в$ = (47387,50 + 5050,90) · 50 / 100% = 26219,20 грн.

4.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.16)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв} = 100\%$.

$В_{нзв}$ = (47387,50 + 5050,90) · 100 / 100% = 52438,40 грн.

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія організації змагань з кібербезпеки» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.17)$$

$В_{заг}$= 47387,50 +5050,90 +5243,84 +12690,09 +4462,70 +2545,40 +41630,60 + 55,00 + 3881,67 +870,48 +0,00 +0,00 +26219,20 +52438,40 = 202475,77 грн.

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.18)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,95.

ЗВ = 202475,77 / 0,95 = 213132,39 грн.

4.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія організації змагань з кібербезпеки» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку основу майбутнього економічного ефекту будуть формувати:

ΔN – збільшення кількості споживачів яким надається відповідна інформаційна послуга у періоди часу, що аналізуються;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	100	150	175	150

N – кількість споживачів яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки, приймемо 5500 осіб;

$Ц_б$ – вартість послуги у році до впровадження інформаційної системи, приймемо 1000,00 грн;

$\pm\Delta Ц_о$ – зміна вартості послуги від впровадження результатів, приймемо зростання на +431,00 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [35].

$$\Delta П_i = (\pm\Delta Ц_о \cdot N + Ц_о \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (4.19)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =40%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1$ =(431,00·5500,00+1431,00·100)·0,83·0,4·(1-0,18/100%)=684302,46 грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2$ =(431,00·5500,00+1431,00·250)·0,83·0,4·(1-0,18/100%)=742738,78 грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3$ =(431,00·5500,00+1431,00·425)·0,83·0,4·(1-0,18/100%)=810914,48 грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4$ =(431,00·5500,00+1431,00·575)·0,83·0,4·(1-0,18/100%)=869350,80 грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{4.20}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,23;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$ПП$ =684302,46/(1+0,23)^1+742738,78/(1+0,23)^2+810914,48/(1+0,23)^3+
+869350,80/(1+0,23)^4=556343,47+490937,13+435772,40+379817,19=1862870,18 грн.

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{4.21}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2,1;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 213132,39 грн.

$PV = k_{інв} \cdot ЗВ$= 2,1 · 213132,39 = 447578,02 грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{4.22}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 1862870,18 грн;

PV – теперішня вартість початкових інвестицій, 447578,02 грн.

$E_{абс} = ПП - PV$ = 1862870,18 - 447578,02 = 1415292,16 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{4.23}$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 1415292,16 грн;

PV – теперішня вартість початкових інвестицій, 447578,02 грн;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1 = (1+1415292,16/447578,02)^{1/4} = 0,43.$$

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \tag{4.24}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,26.

$\tau_{мін}$ = 0,1+0,26 = 0,36 < 0,43 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія організації змагань з кібербезпеки» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \tag{4.25}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,43 = 2,33 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія організації змагань з кібербезпеки»

становить 39,7 бали, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

При оцінюванні за технічними параметрами, згідно узагальненого коефіцієнту якості розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 4,23 рази.

Також термін окупності становить 2,33 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія організації змагань з кібербезпеки».

ВИСНОВКИ

Під час виконання магістерської кваліфікаційної роботи виконано всі поставленні завдання. Проведено аналіз підходів до організації змагань з кібербезпеки. Було проаналізована їх структура, додатково аналіз та порівняння існуючих рішень для організації змагань з кібербезпеки. В процесі порівняння була проведена оцінка сучасних рішень на відповідність інформаційній технології. В результаті стало зрозуміло, що сучасні існуючі рішення не в повній мірі відповідають необхідним критеріям оцінки інформаційної технології організації змагання з кібербезпеки. Прийнято рішення про власну розробку програмної реалізації системи, що в повній мірі відповідає та покриває процеси інформаційної технології.

Також, було розроблено саму інформаційну технології та описано її процеси на рівні організатора та учасника. Розглянуто методики розробки практичних завдань в рамках інформаційної технології. Розроблено архітектуру програмного засобу та виконано розбиття серверної частини на окремі модулі.

Виконано обґрунтування обраних інструментальних засобів розробки системи. Розроблено програмний засіб інформаційної технології у вигляді веб-додатку, що складається з серверної частини та клієнтських частин організатора та учасника. Також, виконана розробка практичних завдань відповідно до описаних методів. Розглянуто сценарії виконання завдань в категоріях «веб» та «форензіка» від лиця учасника. На базі розробленої системи та завдання проведено змагання з кібербезпеки, після чого виконано аналіз його результатів.

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія організації змагань з кібербезпеки» становить 39,7 бали, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коркошко В., Куперштейн Л. Платформа для проведення змагань з кібербезпеки. URL: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2021/paper/view/12193.
2. An Empirical Survey of Functions and Congurations of Open-Source Capture the Flag (CTF) Environments. URL: https://marialeitner.org/wp-content/papercite-data/pdf/kucekl_empirical_2020.pdf (дата звернення 10.09.2022).
3. PHDays CTF. URL: https://ctf.phdays.com/ (дата звернення 12.09.2022).
4. Vinnitsia Linuxation. URL: https://linuxation.in.ua/ (дата звернення 12.09.2022).
5. Cloud Object Storage | Store & Retrieve Data Anywhere | Amazon Simple Storage Service (S3). URL: https://aws.amazon.com/s3/?nc1=h_ls (дата звернення 12.09.2022).
6. CTFd Docs. URL: https://docs.ctfd.io/ (дата звернення 10.10.2022).
7. HTB Capture The Flag Platform | Find & Play Hacking CTFs!. URL: https://ctf.hackthebox.com (дата звернення 10.10.2022)
8. CVE. URL: https://www.cve.org (дата звернення 05.10.2022).
9. Освітньо-професійна програма – 125 кібербезпека (бакалавр). URL: https://vntu.edu.ua/uploads/2022/b_125_bez_inf_kom_sys_OPP_2021.pdf (дата звернення 08.10.2022).
10. CVE Details. URL: https://www.cvedetails.com/ (дата звернення 09.10.2022).
11. Wireshark . URL: https://www.wireshark.org/ (дата звернення 09.10.2022).
12. End-to-end API Security for Cloud-Native Applications Walaram. URL: https://www.wallarm.com/ (дата звернення 12.10.2022).
13. GraphQL Batching Attack. URL: https://lab.wallarm.com/graphql-batching-attack/ (дата звернення 13.10.2022).
14. GraphQL | A query language for your API. URL: https://graphql.org (дата звернення 13.10.2022).

15. One Time Password (OTP meaning with examples). URL: https://www.thalesgroup.com/en/markets/digital-identity-and-security/technology/otp (дата звернення 13.10.2022).
16. Documentation | Node.js. URL: https://nodejs.org/en/docs/ (дата звернення 18.10.2022).
17. What is REST - REST API Tutorial. URL: https://restfulapi.net (дата звернення 18.10.2022).
18. Docker Documentation | Docker Documentation. URL: https://docs.docker.com/ (дата звернення 18.10.2022).
19. PHP: Documentation. URL: https://www.php.net/docs.php. (дата звернення 27.11.2022).
20. PHP Standards Recommendations - PHP-FIG. URL: https://www.php-fig.org/psr/ (дата звернення 28.11.2022).
21. Composer. URL: https://getcomposer.org/doc/ (дата звернення 28.11.2022).
22. Symfony Documentation. URL: https://symfony.com/doc/current/index.html (дата звернення 24.11.2022).
23. Installation - Laravel - The PHP Framework For Web Artisans. URL: https://laravel.com/docs/9.x/installation (дата звернення 26.11.2022).
24. What is an ORM – The Meaning of Object Relational Mapping Database Tools. URL: https://www.freecodecamp.org/news/what-is-an-orm-the-meaning-of-object-relational-mapping-database-tools/ (дата звернення 01.12.2022).
25. Doctrine: PHP Open Source Project. URL: https://www.doctrine-project.org/ (дата звернення 01.12.2022).
26. 2.4. Data Mapper — DesignPatternsPHP 1.0 documentation. URL: https://designpatternsphp.readthedocs.io/en/latest/Structural/DataMapper/README.html (дата звернення 05.12.2022).
27. The Active Record Design Pattern. URL: https://researchhubs.com/post/computing/web-application/the-active-record-design-pattern.html (дата звернення 05.12.2022).

28. Clean Coder Blog. URL: https://blog.cleancoder.com/uncle-bob/2020/10/18/Solid-Relevance.html (дата звернення 05.12.2022).
29. PostgreSQL: Documentation. URL: https://www.postgresql.org/docs/ (дата звернення 07.12.2022).
30. Docs & Demos - Documentation | PhpStorm. URL: https://www.jetbrains.com/phpstorm/documentation/ (дата звернення 07.12.2022).
31. NGINX Rate Limiting. URL: https://www.nginx.com/blog/rate-limiting-nginx (дата звернення 10.12.2022).
32. npm Docs. URL: https://docs.npmjs.com (дата звернення 10.12.2022).
33. Supervisor: A Process Control System — Supervisor 4.2.5 documentation. URL: http://supervisord.org/ (дата звернення 12.12.2022).
34. ZIP. URL: https://docs.fileformat.com/compression/zip/ (дата звернення 15.12.2022).
35. Dockerfile reference | Docker Documentation. URL: https://docs.docker.com/engine/reference/builder/ (дата звернення 05.10.2022).
36. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
37. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток А

ПРОТОКОЛ ПЕРЕВІРКИ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія організації змагань з кібербезпеки
Автор роботи: Коркошко Віктор Русланович
Тип роботи: магістерська кваліфікаційна робота

Підрозділ кафедра захисту інформації ФІТКІ
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність – 97.9%. Схожість – 2.1%.

Аналіз звіту подібності (відмітити потрібне):

√ 1. Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ 2. Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ 3. Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Каплун В. А.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Коркошко В.Р.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Кушнірейчик І.Ю.
(підпис) (прізвище, ініціали)

Додаток Б

Текст програми

AuthServiceProvider.php

```
<?php

namespace App\Providers;

// use Illuminate\Support\Facades\Gate;
use Illuminate\Foundation\Support\Providers\AuthServiceProvider as ServiceProvider;

class AuthServiceProvider extends ServiceProvider
{
    /**
     * The model to policy mappings for the application.
     *
     * @var array<class-string, class-string>
     */
    protected $policies = [
        // 'App\Models\Model' => 'App\Policies\ModelPolicy',
    ];

    /**
     * Register any authentication / authorization services.
     *
     * @return void
     */
    public function boot()
    {
        $this->registerPolicies();

        //
    }
}
```

RouteServiceProvider.php

```
<?php

namespace App\Providers;

use Illuminate\Cache\RateLimiting\Limit;
use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\RateLimiter;
use Illuminate\Support\Facades\Route;

class RouteServiceProvider extends ServiceProvider
{
    /**
     * The path to the "home" route for your application.
     *
     * Typically, users are redirected here after authentication.
     *
     * @var string
     */
    public const HOME = '/home';

    /**
     * Define your route model bindings, pattern filters, and other route configuration.
     *
     * @return void
     */
    public function boot()
    {
        $this->configureRateLimiting();

        $this->routes(function () {
            Route::middleware('api')
                ->prefix('api')
                ->group(base_path('routes/api.php'));

            Route::middleware('web')
                ->group(base_path('routes/web.php'));
        });
    }
```

```
    /**
     * Configure the rate limiters for the application.
     *
     * @return void
     */
    protected function configureRateLimiting()
    {
        RateLimiter::for('api', function (Request $request) {
            return Limit::perMinute(60)->by($request->user()?->id ?: $request->ip());
        });
    }
}
```

BroadcastServiceProvider.php

```
<?php

namespace App\Providers;

use Illuminate\Support\Facades\Broadcast;
use Illuminate\Support\ServiceProvider;

class BroadcastServiceProvider extends ServiceProvider
{
    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        Broadcast::routes();

        require base_path('routes/channels.php');
    }
}
```

EventServiceProvider.php

```
<?php

namespace App\Providers;

use Illuminate\Auth\Events\Registered;
use Illuminate\Auth\Listeners\SendEmailVerificationNotification;
use Illuminate\Foundation\Support\Providers\EventServiceProvider as ServiceProvider;
use Illuminate\Support\Facades\Event;

class EventServiceProvider extends ServiceProvider
{
    /**
     * The event to listener mappings for the application.
     *
     * @var array<class-string, array<int, class-string>>
     */
    protected $listen = [
        Registered::class => [
            SendEmailVerificationNotification::class,
        ],
    ];

    /**
     * Register any events for your application.
     *
     * @return void
     */
    public function boot()
    {
        //
    }

    /**
     * Determine if events and listeners should be automatically discovered.
     *
     * @return bool
     */
    public function shouldDiscoverEvents()
    {
        return false;
```

```
    }
}
```

Handler.php

```
<?php

namespace App\Exceptions;

use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
use Throwable;

class Handler extends ExceptionHandler
{
    /**
     * A list of exception types with their corresponding custom log levels.
     *
     * @var array<class-string<\Throwable>, \Psr\Log\LogLevel::*>
     */
    protected $levels = [
        //
    ];

    /**
     * A list of the exception types that are not reported.
     *
     * @var array<int, class-string<\Throwable>>
     */
    protected $dontReport = [
        //
    ];

    /**
     * A list of the inputs that are never flashed to the session on validation exceptions.
     *
     * @var array<int, string>
     */
    protected $dontFlash = [
        'current_password',
        'password',
        'password_confirmation',
    ];

    /**
     * Register the exception handling callbacks for the application.
     *
     * @return void
     */
    public function register()
    {
        $this->reportable(function (Throwable $e) {
            //
        });
    }
}
```

Team.php

```
<?php

namespace App\Team\Models;

use App\Event\Models\Event;
use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;
use Illuminate\Database\Eloquent\Relations\HasMany;

/**
 * @property string $uuid
 * @property string $name
 * @property Member $leader;
 * @property Collection<Member> $members
 * @property Collection<Event> $events
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 */
```

```
final class Team extends Model
{
    use HasUuids;

    protected $primaryKey = 'uuid';

    public function isLeader(Member $member): bool
    {
        return $this->leader->uuid === $member->uuid;
    }

    public function leader(): BelongsTo
    {
        return $this->belongsTo(Member::class);
    }

    public function members(): HasMany
    {
        return $this->hasMany(Member::class);
    }
}
```

Member.php

```
<?php

namespace App\Team\Models;

use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

/**
 * @property string $uuid
 * @property string $name
 * @property string $email
 * @property Team $team;
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 */
final class Member extends Model
{
    use HasUuids;

    protected $with = ['team'];

    protected $primaryKey = 'uuid';

    public function isTeamLeader(): bool
    {
        return $this->team->isLeader($this);
    }

    public function team(): BelongsTo
    {
        return $this->belongsTo(Team::class);
    }
}
```

Invite.php

```
<?php

namespace App\Team\Models;

use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

/**
 * @property string $uuid
 * @property string $email
 * @property Team $team;
 * @property CarbonImmutable $sent_at
 * @property CarbonImmutable $accepted_at
 * @property CarbonImmutable $deleted_at
 *
```

```
 */
final class Invite extends Model
{
    use HasUuids;
    use SoftDeletes;

    protected $primaryKey = 'uuid';
}
```

RedirectIfAuthenticated.php

```
<?php

namespace App\Http\Middleware;

use App\Providers\RouteServiceProvider;
use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class RedirectIfAuthenticated
{
    /**
     * Handle an incoming request.
     *
     * @param  \Illuminate\Http\Request  $request
     *                 @param                                    \Closure(\Illuminate\Http\Request):
(\Illuminate\Http\Response|\Illuminate\Http\RedirectResponse)  $next
     * @param  string|null  ...$guards
     * @return \Illuminate\Http\Response|\Illuminate\Http\RedirectResponse
     */
    public function handle(Request $request, Closure $next, ...$guards)
    {
        $guards = empty($guards) ? [null] : $guards;

        foreach ($guards as $guard) {
            if (Auth::guard($guard)->check()) {
                return redirect(RouteServiceProvider::HOME);
            }
        }

        return $next($request);
    }
}
```

TrimStrings.php

```
<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\TrimStrings as Middleware;

class TrimStrings extends Middleware
{
    /**
     * The names of the attributes that should not be trimmed.
     *
     * @var array<int, string>
     */
    protected $except = [
        'current_password',
        'password',
        'password_confirmation',
    ];
}
```

Authenticate.php

```
<?php

namespace App\Http\Middleware;

use Illuminate\Auth\Middleware\Authenticate as Middleware;

class Authenticate extends Middleware
{
    /**
     * Get the path the user should be redirected to when they are not authenticated.
     *
```

```
     * @param  \Illuminate\Http\Request  $request
     * @return string|null
     */
    protected function redirectTo($request)
    {
        if (! $request->expectsJson()) {
            return route('login');
        }
    }
}
```

GetChallengeAction.php

```
<?php

namespace App\Http\Controllers\Api\Challenge;

use App\Challenge\Models\Challenge;
use Illuminate\Http\Request;

class GetChallengeAction
{
    public function __invoke(Request $request)
    {
        $challenges = Challenge::query()->paginate();

        '';
    }
}
```

Kernel.php

```
<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel
{
    /**
     * The application's global HTTP middleware stack.
     *
     * These middleware are run during every request to your application.
     *
     * @var array<int, class-string|string>
     */
    protected $middleware = [
        // \App\Http\Middleware\TrustHosts::class,
        \App\Http\Middleware\TrustProxies::class,
        \Illuminate\Http\Middleware\HandleCors::class,
        \App\Http\Middleware\PreventRequestsDuringMaintenance::class,
        \Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,
        \App\Http\Middleware\TrimStrings::class,
        \Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,
    ];

    /**
     * The application's route middleware groups.
     *
     * @var array<string, array<int, class-string|string>>
     */
    protected $middlewareGroups = [
        'web' => [
            \App\Http\Middleware\EncryptCookies::class,
            \Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,
            \Illuminate\Session\Middleware\StartSession::class,
            \Illuminate\View\Middleware\ShareErrorsFromSession::class,
            \App\Http\Middleware\VerifyCsrfToken::class,
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],

        'api' => [
            // \Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful::class,
            'throttle:api',
            \Illuminate\Routing\Middleware\SubstituteBindings::class,
        ],
    ];
```

```
    /**
     * The application's route middleware.
     *
     * These middleware may be assigned to groups or used individually.
     *
     * @var array<string, class-string|string>
     */
    protected $routeMiddleware = [
        'auth' => \App\Http\Middleware\Authenticate::class,
        'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
        'auth.session' => \Illuminate\Session\Middleware\AuthenticateSession::class,
        'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
        'can' => \Illuminate\Auth\Middleware\Authorize::class,
        'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
        'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
        'signed' => \App\Http\Middleware\ValidateSignature::class,
        'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
        'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
    ];
}
```

Category.php

```
<?php

namespace App\Challenge\Models;

use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

/**
 * @property string $uuid
 * @property string $name
 * @property string $description
 * @property Collection<Challenge> $challenges
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 */
class Category extends Model
{
    use HasUuids;
    use SoftDeletes;

    public $primaryKey = 'uuid';
}
```

Flag.php

```
<?php

namespace App\Challenge\Models;

use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

/**
 * @property string $uuid
 * @property string $value
 * @property int $cost
 * @property Challenge $challenge
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 */
final class Flag extends Model
{
    use HasUuids;
    use SoftDeletes;

    public $primaryKey = 'uuid';
```

```
}
```

Solve.php

```
<?php

namespace App\Challenge\Models;

use App\Team\Models\Member;
use App\Team\Models\Team;
use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Model;

/**
 * @property string $uuid
 * @property string $flag
 * @property Challenge $challenge
 * @property Member $member;
 * @property Team $team;
 * @property CarbonImmutable $solved_at
 */
final class Solve extends Model
{
    use HasUuids;

    protected $primaryKey = 'uuid';
}
```

Hint.php

```
<?php

namespace App\Challenge\Models;

use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\SoftDeletes;

/**
 * @property string $uuid
 * @property string $name
 * @property string $description
 * @property Challenge $challenge
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 */
class Hint extends Model
{
    use HasUuids;
    use SoftDeletes;

    public $primaryKey = 'uuid';
}

Status.php
<?php

declare(strict_types=1);

namespace App\Challenge\Models;

enum Status: string
{
    case Draft = 'draft';
    case Active = 'active';
}
```

Challenge.php

```
<?php

namespace App\Challenge\Models;

use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Model;
```

```
use Illuminate\Database\Eloquent\SoftDeletes;

/**
 * @mixin Builder
 *
 * @property string $uuid
 * @property string $name
 * @property string $description
 * @property int $points
 * @property Category $category
 * @property int $category_id
 * @property Collection<Flag> $flags
 * @property Collection<Hint> $hints
 * @property Status $status
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 */
final class Challenge extends Model
{
    use HasUuids;
    use SoftDeletes;

    public $primaryKey = 'uuid';

    public function draft(): self
    {
        $this->status = Status::Draft;

        return $this;
    }
}
```

Attempt.php

```
<?php

namespace App\Challenge\Models;

use App\Team\Models\Member;
use App\Team\Models\Team;
use Carbon\CarbonImmutable;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Model;

/**
 * @property string $uuid
 * @property string $flag
 * @property Challenge $challenge
 * @property Member $member;
 * @property Team $team;
 * @property CarbonImmutable $attempted_at
 */
final class Attempt extends Model
{
    use HasUuids;

    protected $primaryKey = 'uuid';
}
```

ChallengeResource.php

```
<?php

namespace App\Challenge\Http\Resources;

use App\Challenge\Models\Challenge;
use Illuminate\Http\Resources\Json\JsonResource;

/**
 * @mixin Challenge
 */
final class ChallengeResource extends JsonResource
{
    public static $wrap = 'challenge';

    /**
     * @param  \Illuminate\Http\Request  $request
     */
    public function toArray($request): array
```

```
    {
        return [
            'uuid'        => $this->uuid,
            'name'        => $this->name,
            'description' => $this->description,
            'points'      => $this->points,
            'createdAt'   => $this->created_at,
        ];
    }
}
```

UpdateChallengeRequest.php

```
<?php

namespace App\Challenge\Http\Requests;

use App\Challenge\Services\UpdateChallengeDto;
use Illuminate\Foundation\Http\FormRequest;

final class UpdateChallengeRequest extends FormRequest
{
    public function rules(): array
    {
        return [
            'name'        => 'required|min:3|max:64',
            'description' => 'required|min:3|max:2048',
            'categoryId'  => 'required|exists:categories,id',
            'points'      => 'required|numeric|min:0',
        ];
    }

    public function dto(): UpdateChallengeDto
    {
        return new UpdateChallengeDto(
            name: $this->get('name'),
            description: $this->get('description'),
            categoryId: $this->get('categoryId'),
            points: $this->get('points')
        );
    }
}
```

CreateChallengeRequest.php

```
<?php

namespace App\Challenge\Http\Requests;

use App\Challenge\Services\CreateChallengeDto;
use Illuminate\Foundation\Http\FormRequest;

final class CreateChallengeRequest extends FormRequest
{
    public function rules(): array
    {
        return [
            'name'        => 'required|min:3|max:64',
            'description' => 'required|min:3|max:2048',
            'categoryId'  => 'required|exists:categories,id',
        ];
    }

    public function dto(): CreateChallengeDto
    {
        return new CreateChallengeDto(
            name: $this->get('name'),
            description: $this->get('description'),
            categoryId: $this->get('categoryId')
        );
    }
}
```

UpdateChallengeController.php

```
<?php

namespace App\Challenge\Http\Controllers;

use App\Challenge\Http\Requests\UpdateChallengeRequest;
use App\Challenge\Http\Resources\ChallengeResource;
```

```
use App\Challenge\Models\Challenge;
use App\Challenge\Services\UpdateChallengeService;
use Spatie\RouteAttributes\Attributes\Route;

class UpdateChallengeController
{
    public function __construct(
        private readonly Challenge $challenges,
        private readonly UpdateChallengeService $challengeService,
    ) {
    }

    #[Route(methods: ['PUT', 'PATCH'], uri: 'challenges/{uuid}', name: 'challenges.update')]
    public function __invoke(UpdateChallengeRequest $request, string $uuid): ChallengeResource
    {
        $challenge = $this->challengeService->update($this->challenges->findOrFail($uuid), $request-
>dto());

        return ChallengeResource::make($challenge);
    }
}
```

CreateChallengeController.php

```
<?php

namespace App\Challenge\Http\Controllers;

use App\Challenge\Http\Requests\CreateChallengeRequest;
use App\Challenge\Http\Resources\ChallengeResource;
use App\Challenge\Services\CreateChallengeService;
use Spatie\RouteAttributes\Attributes\Post;

class CreateChallengeController
{
    public function __construct(
        private readonly CreateChallengeService $challengeService,
    ) {
    }

    #[Post(uri: 'challenges', name: 'challenges.create')]
    public function __invoke(CreateChallengeRequest $request): ChallengeResource
    {
        $challenge = $this->challengeService->create($request->dto());

        return ChallengeResource::make($challenge);
    }
}
```

CreateChallengeService.php

```
<?php

declare(strict_types=1);

namespace App\Challenge\Services;

use App\Challenge\Models\Challenge;

final class CreateChallengeService
{
    public function create(CreateChallengeDto $createChallengeDto): Challenge
    {
        $challenge = new Challenge();
        $challenge->name = $createChallengeDto->name;
        $challenge->description = $createChallengeDto->description;
        $challenge->category_id = $createChallengeDto->categoryId;
        $challenge->save();

        return $challenge;
    }
}
```

UpdateChallengeService.php

```
<?php

declare(strict_types=1);

namespace App\Challenge\Services;
```

```
use App\Challenge\Models\Challenge;

final class UpdateChallengeService
{
    public function update(Challenge $challenge, UpdateChallengeDto $updateChallengeDto): Challenge
    {
        $challenge->name = $updateChallengeDto->name;
        $challenge->description = $updateChallengeDto->description;
        $challenge->category_id = $updateChallengeDto->categoryId;
        $challenge->points = $updateChallengeDto->points;
        $challenge->save();

        return $challenge;
    }
}
```

CreateChallengeDto.php

```
<?php

declare(strict_types=1);

namespace App\Challenge\Services;

final class CreateChallengeDto
{
    public function __construct(
        public readonly string $name,
        public readonly string $description,
        public readonly int $categoryId,
    ) {
    }
}
```

UpdateChallengeDto.php

```
<?php

declare(strict_types=1);

namespace App\Challenge\Services;

final class UpdateChallengeDto
{
    public function __construct(
        public readonly string $name,
        public readonly string $description,
        public readonly int $categoryId,
        public readonly int $points,
    ) {
    }
}
```

Event.php

```
<?php

namespace App\Event\Models;

use App\Challenge\Models\Category;
use App\Challenge\Models\Challenge;
use App\Challenge\Models\Flag;
use App\Challenge\Models\Hint;
use Carbon\CarbonImmutable;
use Database\Factories\EventFactory;
use DomainException;
use Illuminate\Database\Eloquent\Builder;
use Illuminate\Database\Eloquent\Collection;
use Illuminate\Database\Eloquent\Concerns\HasUuids;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsToMany;
use Illuminate\Database\Eloquent\Relations\HasMany;

/**
 * @mixin Builder
 *
 * @property string $uuid
 * @property string $name
 * @property string $description
```

```
 * @property bool $manually_start
 * @property Collection<Challenge> $challenges
 * @property Collection<Participant> $participants
 * @property Status $status
 * @property CarbonImmutable $starts_at
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 *
 * @method Builder|static pending()
 * @method Builder|static notManuallyStart()
 */
class Event extends Model
{
    use HasFactory;
    use HasUuids;

    protected $primaryKey = 'uuid';

    public static function newFactory()
    {
        return new EventFactory();
    }

    public function pause(): self
    {
        $this->status = Status::Paused;

        return $this;
    }

    public function start(): self
    {
        $this->status = Status::Started;

        return $this;
    }

    public function finish(): self
    {
        $this->status = Status::Finished;

        return $this;
    }

    public function finishDraft(): self
    {
        if ($this->status !== Status::Draft) {
            throw new DomainException('Only a draft event can be moved to pending');
        }

        $this->status = Status::Pending;

        return $this;
    }

    public function draft(): self
    {
        $this->status = Status::Draft;

        return $this;
    }

    public function isPaused(): bool
    {
        return $this->status === Status::Paused;
    }

    public function isDrafted(): bool
    {
        return $this->status === Status::Draft;
    }

    public function scopePending(Builder $builder): void
    {
        $builder->where('status', Status::Pending);
    }

    public function scopeNotManuallyStart(Builder $builder): void
    {
```

```
        $builder->where('manually_start', false);
    }

    public function challenges(): BelongsToMany
    {
        return $this->belongsToMany(Challenge::class);
    }

    public function participants(): BelongsToMany
    {
        return $this->belongsToMany(Participant::class);
    }
}
```

Status.php

```
<?php

namespace App\Event\Models;

enum Status: string
{
    case Draft = 'draft';
    case Pending = 'pending';
    case Started = 'started';
    case Paused = 'paused';
    case Finished = 'finished';
}
```

Participant.php

```
<?php

namespace App\Event\Models;

use App\Team\Models\Member;
use App\Team\Models\Team;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\MorphTo;

/**
 * @property string $uuid
 * @property string $name
 * @property Member $leader;
 * @property Collection<Member> $members
 * @property Collection<Event> $events
 * @property CarbonImmutable $created_at
 * @property CarbonImmutable $updated_at
 * @property CarbonImmutable $deleted_at
 */
final class Participant extends Model
{
    protected $primaryKey = 'uuid';

    public function team(): MorphTo
    {
        return $this->morphTo(Team::class);
    }

    public function member(): MorphTo
    {
        return $this->morphTo(Member::class);
    }
}
```

EventResource.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Resources;

use App\Event\Models\Event;
use Illuminate\Http\Resources\Json\JsonResource;

/**
 * @mixin Event
 */
final class EventResource extends JsonResource
{
```

```
    public static $wrap = 'event';

    public function toArray($request): array
    {
        return [
            'uuid'          => $this->uuid,
            'name'          => $this->name,
            'description'   => $this->description,
            'status'        => $this->status,
            'startsAt'      => $this->starts_at,
            'manuallyStart' => $this->manually_start,
            'createdAt'     => $this->created_at,
        ];
    }
}
```

CreateEventRequest.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Requests;

use App\Event\Services\CreateEventDto;
use Illuminate\Foundation\Http\FormRequest;

final class CreateEventRequest extends FormRequest
{
    public function rules(): array
    {
        return [
            'name'          => 'required|min:3|max:64',
            'description'   => 'required|min:3|max:2048',
            'manuallyStart' => 'required|bool',
            'startsAt'      => 'sometimes|date',
        ];
    }

    public function dto(): CreateEventDto
    {
        return new CreateEventDto(
            name: $this->get('name'),
            description: $this->get('description'),
            manuallyStart: $this->get('manuallyStart'),
            startsAt: $this->get('startsAt'),
        );
    }
}
```

UpdateEventRequest.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Requests;

use App\Event\Services\CreateEventDto;
use App\Event\Services\UpdateEventDto;
use Illuminate\Foundation\Http\FormRequest;

final class UpdateEventRequest extends FormRequest
{
    public function rules(): array
    {
        return [
            'name'          => 'required|min:3|max:64',
            'description'   => 'required|min:3|max:2048',
            'challenges'    => 'required|array|min:1',
            'challenges.*'  => 'required|exists:challenges,uuid',
            'manuallyStart' => 'required|bool',
            'startsAt'      => 'sometimes|date',
        ];
    }

    public function dto(): UpdateEventDto
    {
        return new UpdateEventDto(
            name: $this->get('name'),
```

```
            description: $this->get('description'),
            manuallyStart: $this->get('manuallyStart'),
            challenges: $this->get('challenges'),
            startsAt: $this->get('startsAt'),
        );
    }
}
```

IndexEventController.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Controllers;

use App\Event\Http\Resources\EventResource;
use App\Event\Models\Event;
use Illuminate\Http\Resources\Json\AnonymousResourceCollection;
use Spatie\RouteAttributes\Attributes\Get;

final class IndexEventController
{
    public function __construct(
        private readonly Event $events,
    ) {
    }

    #[Get(uri: 'events', name: 'events.events')]
    public function __invoke(): AnonymousResourceCollection
    {
        $events = $this->events->paginate();

        AnonymousResourceCollection::wrap('events');

        return EventResource::collection($events);
    }
}
```

CreateEventController.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Controllers;

use App\Event\Http\Requests\CreateEventRequest;
use App\Event\Http\Resources\EventResource;
use App\Event\Services\CreateEventService;
use Spatie\RouteAttributes\Attributes\Post;

final class CreateEventController
{
    public function __construct(
        private readonly CreateEventService $eventService,
    ) {
    }

    #[Post(uri: 'events', name: 'events.create')]
    public function __invoke(CreateEventRequest $request): EventResource
    {
        $event = $this->eventService->create($request->dto());

        return EventResource::make($event);
    }
}
```

ShowEventController.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Controllers;

use App\Event\Http\Resources\EventResource;
use App\Event\Models\Event;
use Spatie\RouteAttributes\Attributes\Get;

final class ShowEventController
```

```
{
    public function __construct(
        private readonly Event $events,
    ) {
    }

    #[Get(uri: 'events/{uuid}', name: 'events.show')]
    public function __invoke(string $uuid): EventResource
    {
        $event = $this->events->findOrFail($uuid);

        return EventResource::make($event);
    }
}
```

FinishDraftController.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Controllers;

use App\Event\Http\Resources\EventResource;
use App\Event\Models\Event;
use App\Event\Services\FinishDraftEventService;
use Spatie\RouteAttributes\Attributes\Post;

final class FinishDraftController
{
    public function __construct(
        private readonly Event $events,
        private readonly FinishDraftEventService $eventService,
    ) {
    }

    #[Post(uri: 'events/:uuid/finish-draft', name: 'events.finish_draft')]
    public function __invoke(string $uuid): EventResource
    {
        $event = $this->eventService->finishDraft($this->events->findOrFail($uuid));

        return EventResource::make($event);
    }
}
```

UpdateEventController.php

```
<?php

declare(strict_types=1);

namespace App\Event\Http\Controllers;

use App\Event\Http\Requests\UpdateEventRequest;
use App\Event\Http\Resources\EventResource;
use App\Event\Models\Event;
use App\Event\Services\UpdateEventService;
use Spatie\RouteAttributes\Attributes\Route;

final class UpdateEventController
{
    public function __construct(
        private readonly Event $events,
        private readonly UpdateEventService $eventService,
    ) {
    }

    #[Route(methods: ['PUT', 'PATCH'], uri: 'events/:uuid', name: 'events.update')]
    public function __invoke(UpdateEventRequest $request, string $uuid): EventResource
    {
        $event = $this->eventService->update($this->events->findOrFail($uuid), $request->dto());

        return EventResource::make($event);
    }
}
```

UpdateEventService.php

```
<?php

declare(strict_types=1);
```

```
namespace App\Event\Services;

use App\Event\Models\Event;
use Carbon\CarbonImmutable;

final class UpdateEventService
{
    public function update(Event $event, UpdateEventDto $updateEventDto): Event
    {
        $event->name = $updateEventDto->name;
        $event->description = $updateEventDto->description;

        if ($updateEventDto->startsAt) {
            $event->starts_at = CarbonImmutable::parse($updateEventDto->startsAt);
        }

        $event->challenges()->sync($updateEventDto->challenges);
        $event->save();

        return $event;
    }
}
```

FinishDraftEventService.php

```
<?php

declare(strict_types=1);

namespace App\Event\Services;

use App\Event\Models\Event;

final class FinishDraftEventService
{
    public function finishDraft(Event $event): Event
    {
        $event->finishDraft();
        $event->save();

        return $event;
    }
}
```

StartPendingEventService.php

```
<?php

declare(strict_types=1);

namespace App\Event\Services;

use App\Event\Models\Event;

final class StartPendingEventService
{
    public function __construct(
        private readonly Event $events,
    ) {
    }

    public function process(): void
    {
        $events = $this->events->pending()->notManuallyStart()->get();

        foreach ($events as $event) {
            $event->start();
            $event->save();
        }
    }
}
```

CreateEventService.php

```
<?php

declare(strict_types=1);

namespace App\Event\Services;
```

```
use App\Event\Models\Event;
use Carbon\CarbonImmutable;

final class CreateEventService
{
    public function create(CreateEventDto $createEventDto): Event
    {
        $event = new Event();
        $event->name = $createEventDto->name;
        $event->description = $createEventDto->description;

        if ($createEventDto->startsAt) {
            $event->starts_at = CarbonImmutable::parse($createEventDto->startsAt);
        }

        $event->manually_start = $createEventDto->manuallyStart;
        $event->draft();

        return $event;
    }
}
```

CreateEventDto.php

```
<?php

declare(strict_types=1);

namespace App\Event\Services;

final class CreateEventDto
{
    public function __construct(
        public readonly string $name,
        public readonly string $description,
        public readonly bool $manuallyStart,
        public readonly ?string $startsAt = null,
    ) {
    }
}
```

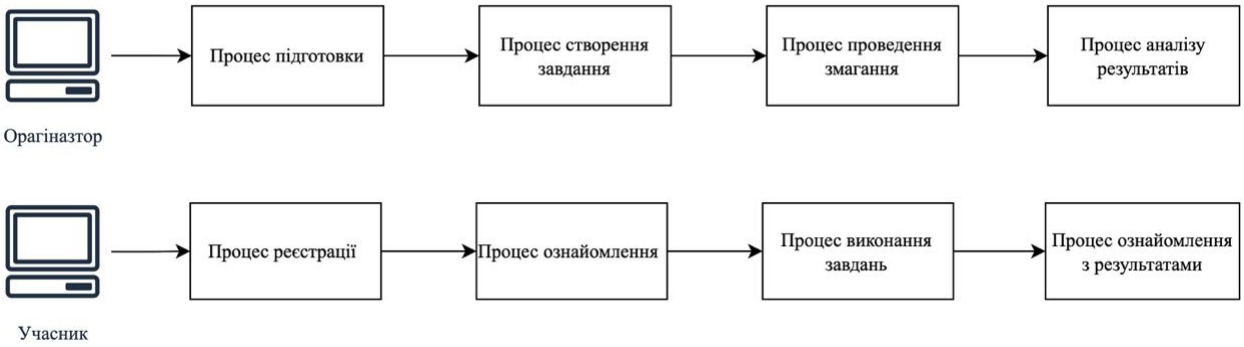
UpdateEventDto.php

```
<?php

declare(strict_types=1);

namespace App\Event\Services;

final class UpdateEventDto
{
    public function __construct(
        public readonly string $name,
        public readonly string $description,
        public readonly bool $manuallyStart,
        public readonly array $challenges,
        public readonly ?string $startsAt = null,
    ) {
    }
}
```

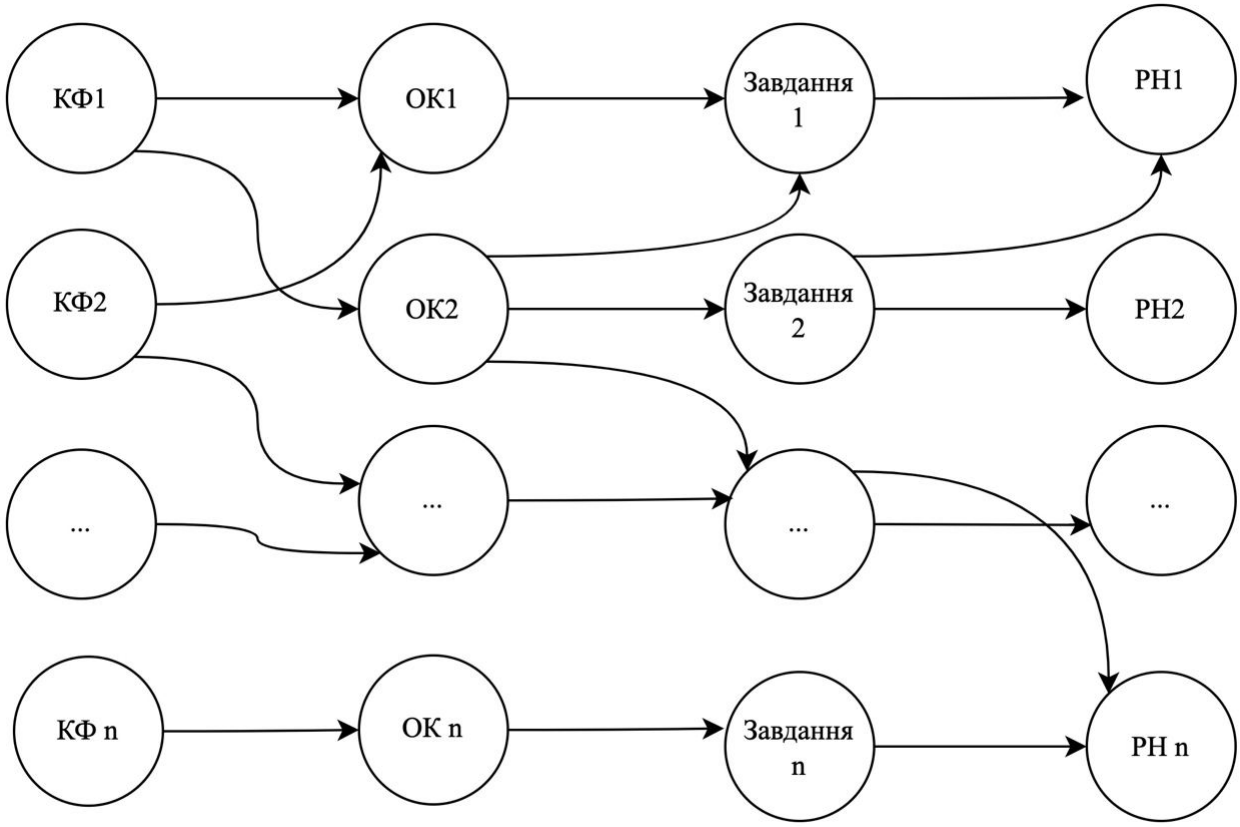
ІЛЮСТРАТИВНА ЧАСТИНА

до магістерської кваліфікаційної роботи
на тему:
«ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ОРГАНІЗАЦІЇ ЗМАГАНЬ З КІБЕРБЕЗПЕКИ»

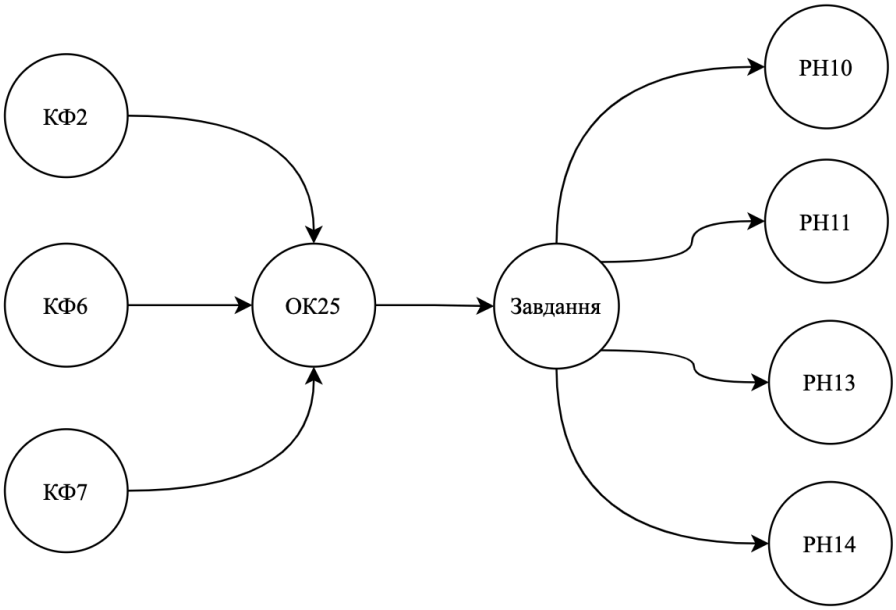
СХЕМИ ПРОЦЕСІВ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ



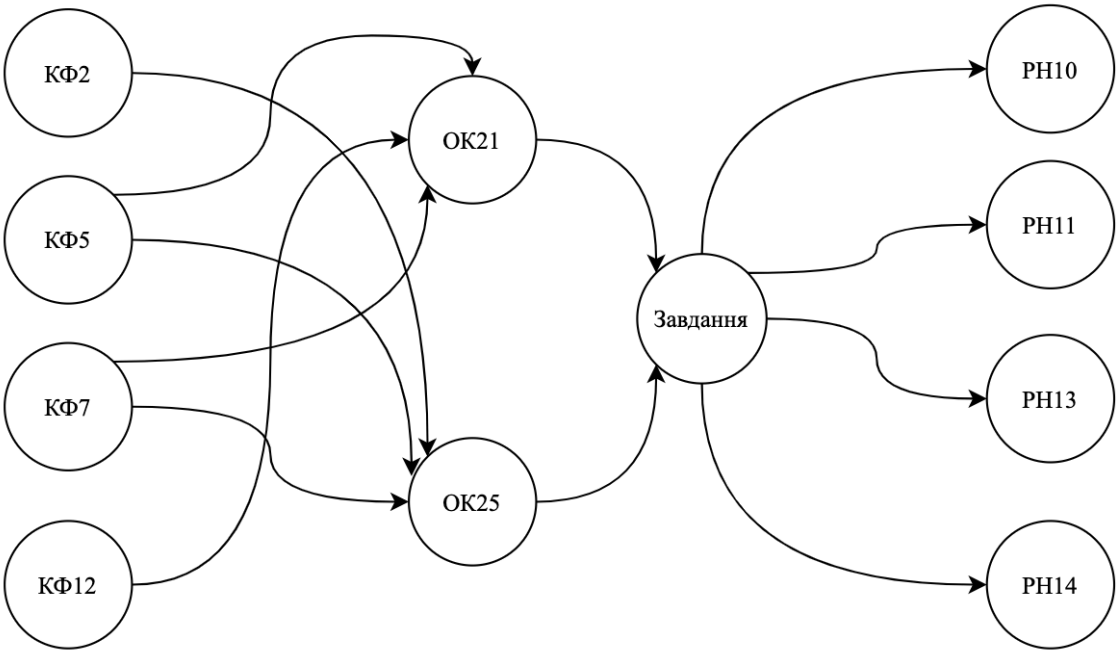
ЗАГАЛЬНА МОДЕЛЬ ВІДПОВІДНОСТІ ЗАВДАНЬ ДО ОПП «БІКС»



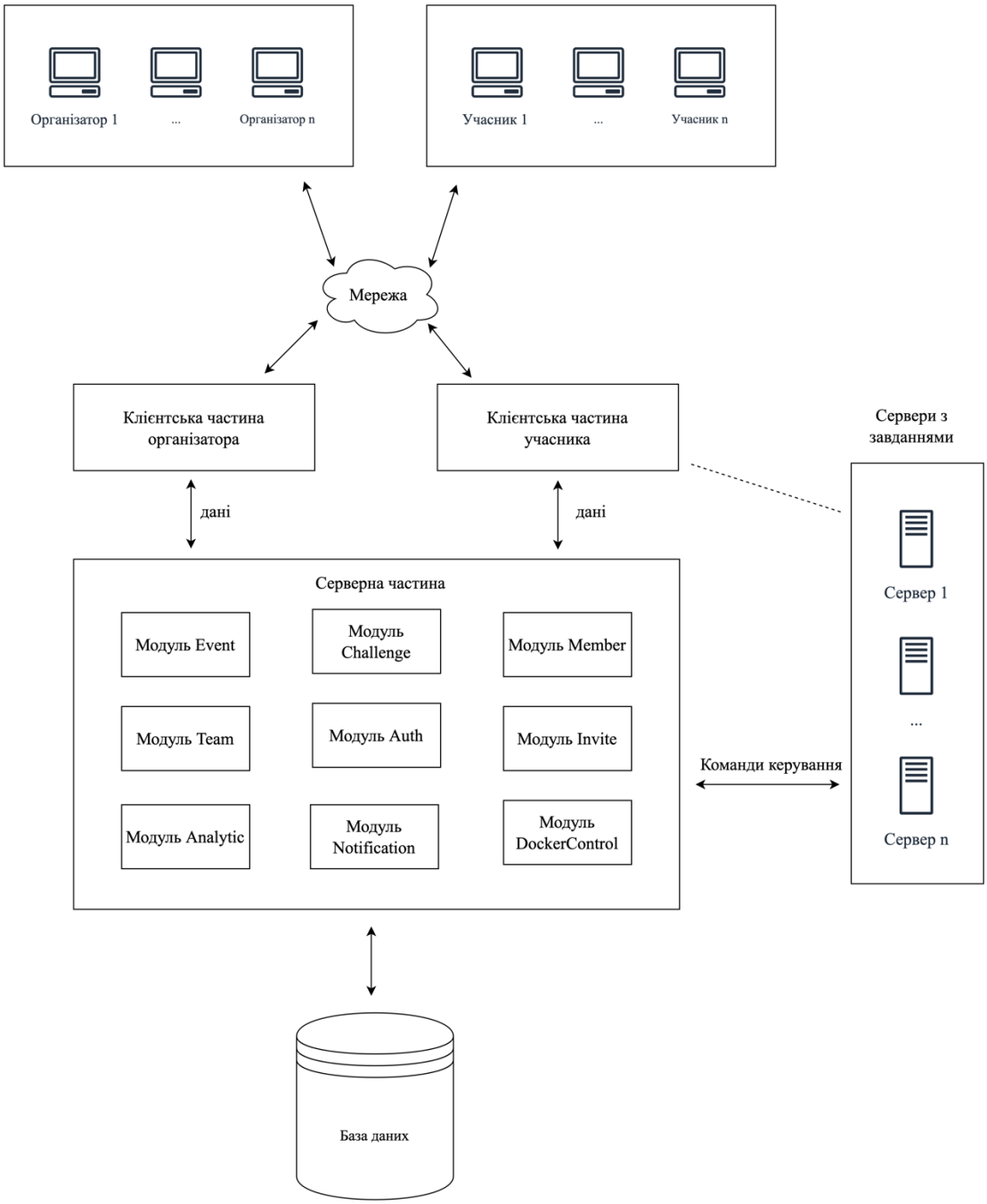
МОДЕЛЬ ВІДПОВІДНОСТІ ЗАВДАННЯ КАТЕГОРІЇ «ФОРЕНЗІКА» ДО ОПП «БІКС»



МОДЕЛЬ ВІДПОВІДНОСТІ ЗАВДАННЯ КАТЕГОРІЇ «ВЕБ» ДО ОПП «БІКС»



АРХІТЕКТУРА СИСТЕМИ



ВИГЛЯД ГОЛОВНОЇ СТОРІНКИ ЗМАГАННЯ