НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

До захисту допущено
Завідувач кафедри

_____ **Дмитро ЛАНДЕ**
(підпис)

«_____»_____2025 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системи, технології та математичні методи кібербезпеки»
спеціальності 125 «Кібербезпека»

на тему: Вибір інструментів тестування кібербезпеки систем електронної торгівлі на основі аналізу Web API

Виконав (-ла) здобувач вищої освіти IV курсу, групи ФБ-11
(шифр групи)

Пташник Юрій Іванович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник Доцент кафедри ІБ, к.т.н., Гальчинський Л. Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2025 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис)
«___»_____2025 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

Пташнику Юрію Івановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Вибір інструментів тестування кібербезпеки систем електронної торгівлі на основі аналізу Web API

науковий керівник роботи Гальчинський Леонід Юрійович к.т.н., доцент, доцент кафедри інформаційної безпеки.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « 26 » ____травня____ 2025 р. № 1761-с

2.Термін подання здобувачем дипломної роботи 12 червня 2025 р.

3. Вихідні дані до роботи:

Літературні джерела, які висвітлюють проблему безпеки API для систем електронної торгівлі, існуючі інструменти тестування кібербезпеки на основі аналізу Web API, алгоритми виконання методів багатокритеріального прийняття рішень.

4. Зміст роботи:

Огляд проблеми безпеки Web API у сфері електронної торгівлі, інструментів тестування кібербезпеки на основі аналізу веб-API, методів багатекритеріального прийняття рішень, використання методу ELECTRE для вибору інструменту тестування безпеки.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація

6. Дата видачі завдання: 07.02.2025

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Формулювання теми та завдання дипломної роботи	07.02.2025 – 19.02.2025	Виконано
2	Вивчення наукової літератури за темою	20.02.2025 - 20.04.2025	Виконано
3	Огляд Web API та їх проблем безпеки у галузі електронної торгівлі	21.04.2025 – 04.05.2025	Виконано
4	Ознайомлення з інструментами тестування безпеки на основі аналізу Web API	05.05.2025 – 18.05.2025	Виконано
5	Дослідження методів багатокритеріального прийняття рішень	19.05.2025 – 25.05.2025	Виконано
6	Виконання процесу вибору інструменту тестування кібербезпеки методом ELECTRE I	26.05.2025 – 01.06.2025	Виконано
7	Оформлення дипломної роботи згідно методичних вказівок	02.06.2025 – 10.06.2025	Виконано
8	Передзахист дипломної роботи	12.06.2025	
9	Захист дипломної роботи	19.06.2025	

Здобувач вищої освіти _____ Юрій ПТАШНИК
(підпис) (власне ім'я, ПРІЗВИЩЕ)

Науковий керівник _____ Леонід ГАЛЬЧИНСЬКИЙ
(підпис) (власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Обсяг роботи 65 сторінок, 14 ілюстрацій, 7 таблиць, 1 додаток, 33 джерела літератури.

Об’єкт дослідження: кібербезпека у сфері електронної торгівлі.

Предмет дослідження: Web API.

Мета дослідження: аналіз сучасного стану безпеки API у сфері електронної торгівлі, проблеми вибору оптимального інструменту тестування безпеки на основі аналізу веб-API та розробка підходу для її вирішення.

Методи дослідження: аналіз літературних джерел, існуючих інструментів тестування кібербезпеки на основі аналізу API.

Отримані результати: сформовано підхід до вибору інструменту тестування API з використанням методу багатокритеріального прийняття рішень. Обґрунтовано вибір критеріїв оцінки обраних засобів для системи електронної торгівлі. На основі підходу було побудовано матриці вибору та ваг, застосовано метод ELECTRE I для отримання ядра вибору альтернатив та виділено найбільш оптимальний інструмент тестування безпеки Web API.

Результати роботи були представлені на XXIII Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» (14 – 17.05.2025 р., м. Київ, Україна).

Ключові слова: Web API, електронна торгівля, ELECTRE I, інструменти тестування безпеки API

ABSTRACT

The volume of the thesis is 65 pages, 14 illustrations, 7 tables, 1 appendix, 33 sources of literature.

Research object: cybersecurity in the field of e-commerce.

The subject of the research: Web API.

The purpose of the research: to analyze the current state of API security in electronic commerce, the challenges of selecting the optimal security-testing tool based on Web API analysis, and to develop an approach to address them.

Research methods: analysis of literary sources; review and comparison of existing cybersecurity testing tools based on API analysis.

Obtained results: an approach for selecting an API testing tool using a multi-criteria decision-making method has been developed. The criteria for evaluating the chosen tools for an e-commerce system were substantiated. Using this approach, decision and weight matrices were constructed, the ELECTRE I method was applied to derive the core set of alternatives, and the most optimal Web API security-testing tool was identified.

The results of the work were presented at the XXIII All-Ukrainian Scientific and Practical Conference of Students, Postgraduates, and Young Scientists «Theoretical and Applied Problems of Physics, Mathematics, and Computer Science» (14 – 17.05.2025, Kyiv, Ukraine).

Keywords: Web API, e-commerce, ELECTRE I, API security-testing tools

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС – інформаційна система.

API (Application Programming Interface) – прикладний програмний інтерфейс.

HTTP (Hypertext Transfer Protocol) – протокол передачі гіпертексту.

URI (Uniform Resource Identifier) – уніфікований ідентифікатор ресурсів.

URL (Uniform Resource Locator) – уніфікований локатор ресурсів.

SaaS (Software as a Service) – програмне забезпечення як послуга.

CI/CD (Continuous Integration / Continuous Deployment) – неперервна інтеграція / неперервне розгортання.

AI/ML (Artificial Intelligence / Machine Learning) – штучний інтелект / машинне навчання.

CLI (Command-Line Interface) – інтерфейс командного рядка.

MCDM (Multi-Criteria Decision Making) – багатокритеріальне прийняття рішень.

SAST (Static Application Security Testing) – статичне тестування безпеки застосунку.

DAST (Dynamic Application Security Testing) – динамічне тестування безпеки застосунку.

SCA (Software Composition Analysis) – аналіз складу програмного забезпечення.

CSPM (Cloud Security Posture Management) – управління хмарною безпекою.

ВСТУП

Актуальність роботи:

Сучасна сфера електронної торгівлі дедалі більше покладається на Web API для взаємодії між клієнтськими застосунками, внутрішніми сервісам та зовнішніми платформами. Однак, стрімке зростання популярності API у комерційному середовищі супроводжується зростанням кількості цілеспрямованих атак на інтерфейси, які часто стають новим вектором атак для несанкціонованого доступу до конфіденційних даних та бізнес-логіки.

За даними моніторингу трафіку, понад 40% усіх веб-атак спрямовані саме на прикладні програмні інтерфейси, а середні фінансові втрати організацій сягають сотень тисяч доларів. Тому вибір ефективного засобу тестування безпеки Web API електронної торгівлі є надзвичайно актуальним завданням.

Метою дослідження: є аналіз сучасного стану безпеки API у сфері електронної торгівлі, проблеми вибору оптимального інструменту тестування безпеки на основі аналізу веб-API та розробка підходу для її вирішення.

Завдання дослідження:

1. Провести огляд сучасного стану безпеки Web API у сфері електронної торгівлі та визначити ключові загрози.
2. Проаналізуати існуючі інструменти статичного, динамічного та поведінкового тестування безпеки API.
3. Проаналізувати методи багатокритеріального прийняття рішень та обрати найбільш релевантний метод.
4. Сформувати набір критеріїв для оцінки розглянутих інструментів тестування кібербезпеки на основі аналізу веб-API.
5. Застосувати обраний метод для ранжування альтернатив і обґрунтування вибору оптимального інструменту.

Об’єкт дослідження: кібербезпека у сфері електронної торгівлі.

Предмет дослідження: Web API.

Методи дослідження: аналіз літературних джерел, існуючих інструментів тестування кібербезпеки на основі аналізу API.

Новизна одержаних результатів: полягає у інтеграції специфічних для електронної торгівлі критеріїв з методом багатокритеріального прийняття рішень, що забезпечує прийняття обґрунтованого рішення з вибору інструменту тестування безпеки API.

Практичне значення одержаних результатів: Це дослідження спрямоване на забезпеченні підходу до обґрунтованого вибору інструменту тестування безпеки API для розробників, інженерів безпеки та адміністраторів систем, які прагнуть забезпечити належний захист власних прикладних програмних інтерфейсів систем електронної торгівлі.

Апробація результатів роботи: Результати дослідження були представлені на XXIII Всеукраїнській науково-практичній конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики» (14 – 17.05.2025 р., м. Київ, Україна), та підхід був розвинений в подальшій роботі.

1 ОГЛЯД ЛІТЕРАТУРНИХ ДЖЕРЕЛ ТА ІНСТРУМЕНТІВ ТЕСТУВАННЯ КІБЕРБЕЗПЕКИ WEB API

Стрімка цифровізація бізнес-процесів та прагнення до підвищення доступності сервісів призводить до значного збільшення кількості кібератак у світі. Все більше організацій та компаній починаються використовувати інформаційні системи для обробки та зберігання конфіденційної інформації, забезпечуючи користувачам зручний доступ до даних з будь-якого пристрою.

Однак, зручність та легкість доступу до даних створює перед організаціями нові виклики кібербезпеки, оскільки зовнішній доступ до ІС надає зловмисникам можливості до спроб викрадення даних та вчинення протиправних дій. За даними Check Point [1], у третьому кварталі 2024 року середня щотижнева кількість кібератак сягнула історичного максимуму у 1876, що означає зростання у 75% порівняно з тим самим періодом 2023 року та зростання у 15% порівняно з минулим кварталом. Динаміку збільшення середньої кількості атак за кварталами можна побачити на рисунку 1.1.

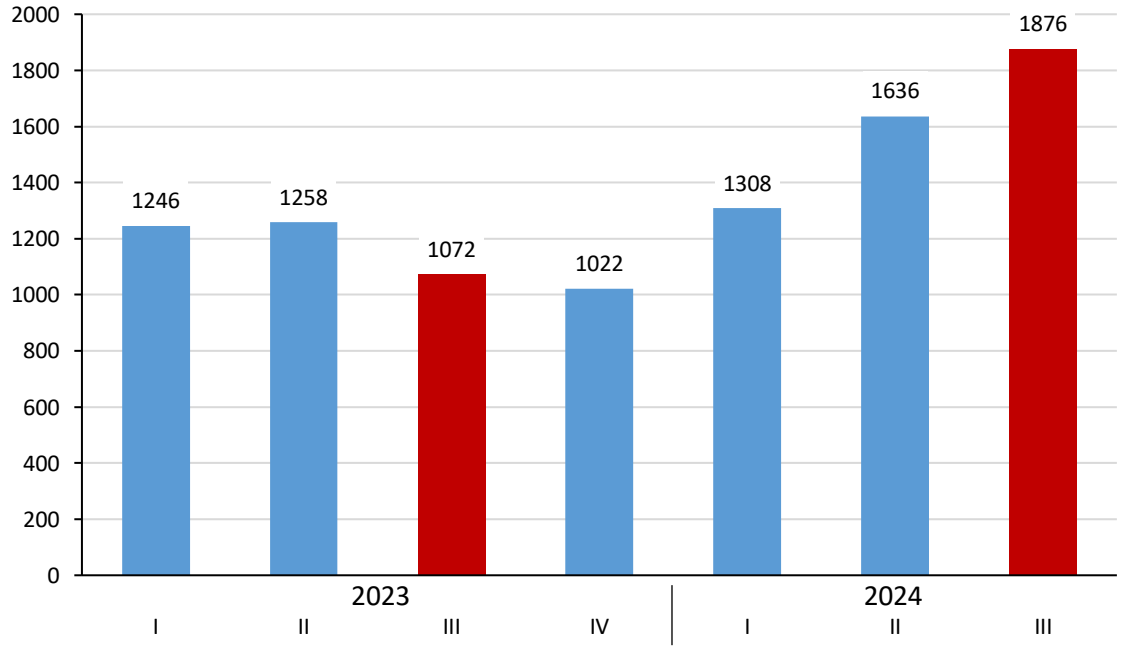

Рисунок 1.1 – Середня кількість щотижневих кібератак за кварталами

Не виключенням у цій статистиці є сфера торгівлі, яка демонструє зростання кількості атак на 53%, порівняно з третім кварталом 2023 року, у результаті

посідаючи восьме місце серед інших сфер за кількістю щотижневих кібератак. На рисунку 1.2 зображено розподіл кібератак за галузями.

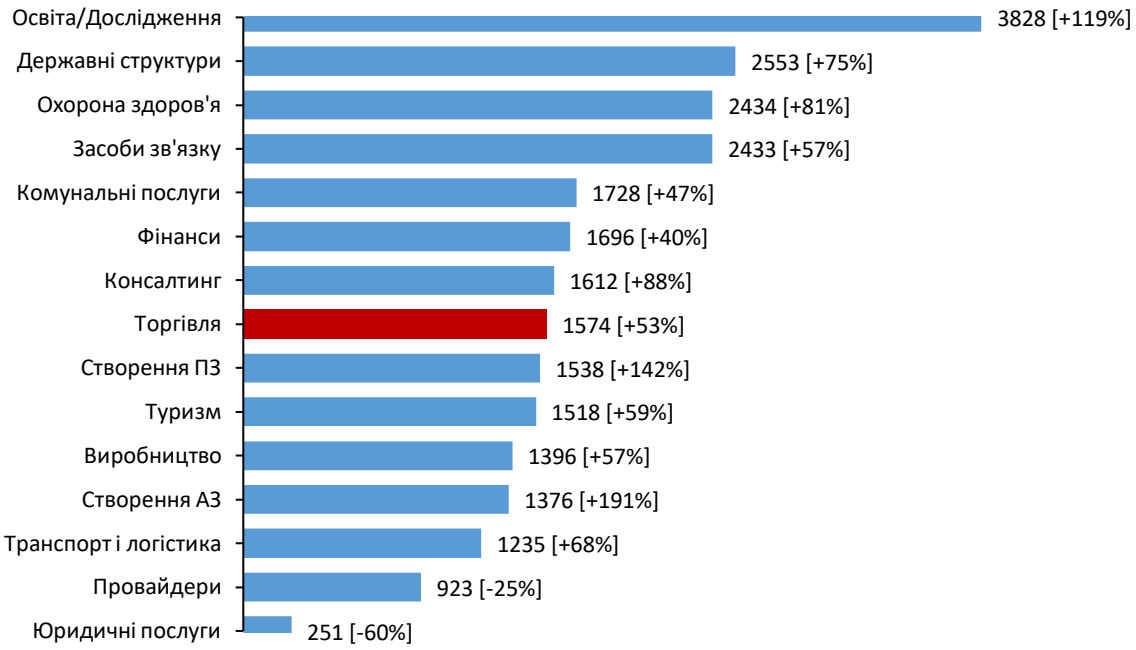

Рисунок 1.2 – Середня кількість щотижневих кібератак за галузями у третьому кварталі 2024 року

Таким чином, можна констатувати глобальну тенденцію до збільшення кількості кібератак у майже всіх сферах людської діяльності в кіберпросторі, зокрема у галузі електронної торгівлі. Ця сфера приваблює зловмисників не лише через наявність у системах конфіденційних та платіжних даних, а й через прагнення державних інституцій до цифровізації процесу закупівель. Електронні торги забезпечують для держави належний рівень прозорості та знижують рівень ризику корупції [2], тому системи електронної торгівлі також стають цілями кібератак, які мають на меті вплив на діяльність державних установ.

Однак, у галузі електронної торгівлі значним викликом є забезпечення належного рівня безпеки прикладних програмних інтерфейсів, які використовуються ІС [3]. Незважаючи на те, що часто API не призначені для загальнодоступного користування, вони зазвичай мають повний доступ до всіх захищених ресурсів та інформації. Такі програмні інтерфейси легко стають цілями зловмисників, а відсутність належного рівня тестування безпеки та бізнес-логіки призводять до загального зростання ризиків, пов’язаних із безпекою API.

1.1 Стан кібербезпеки Web API у сфері електронної торгівлі

Web API є основним елементом сучасних платформ електронної торгівлі, забезпечуючи обмін даними між користувацькими інтерфейсами, внутрішніми компонентами та сторонніми сервісами. Проте відкрита природа прикладних програмних інтерфейсів та складність в управлінні інформаційними потоками, робить API привабливою ціллю для зловмисників.

Згідно з дослідженням Akamai про вплив безпеки API у 2024 році [4], 68% компаній зіткнулися з інцидентами кібербезпеки API, а середні фінансові втрати від них склали 526 531 долар США, 258 815 фунтів стерлінгів та 348 467 євро для організацій, які розташовані у США, Великобританії та Німеччині відповідно. Однак, кібератаки мають не лише фінансовий вплив на роботу бізнесу, на рисунку 1.3 зображено основні наслідки від інцидентів кібербезпеки Web API, отримані з опитування відділів безпеки організацій.

Рисунок 1.3 – Основні наслідки від інцидентів кібербезпеки API

Для організацій розрізнення між справжньою та зловмисною або шахрайською поведінкою веб-API залишається складною задачею. Дана проблема існує через відсутність чіткої видимості ризиків, пов’язаних з прикладними програмними інтерфейсами. Незважаючи на те, що 67% опитаних респондентів стверджують, що

мають повну інвентаризацію усіх їхніх API, лише 29% з даної групи дійсно знають, які Web API повертають конфіденційну інформацію.

Враховуючи вище згадані проблеми, розгортання API без співпраці з командами безпеки та відповідного тестування може нести значні загрози для ІС та організації загалом, наприклад:

- Інтерфейс створено для повернення клієнтських даних без належного контролю авторизації або наявність неправильних налаштувань.
- Веб-API може бути замінений новою версією, але не деактивований. Таким чином, стара версія може знаходитися під зовнішнім впливом.
- Некеровані API можуть бути не виявленні традиційними інструментами.
- Web API може бути використаний шахраями для отримання доступу до облікових записів реальних клієнтів та викрадення грошей.

За результатами опитування відділів безпеки організацій, проведеного дослідницькою командою Akamai, вдалося визначити головні чинники виникнення інцидентів кібербезпеки API у галузі електронної торгівлі. Ці причини та їх розподіл зображено на рисунку 1.4.

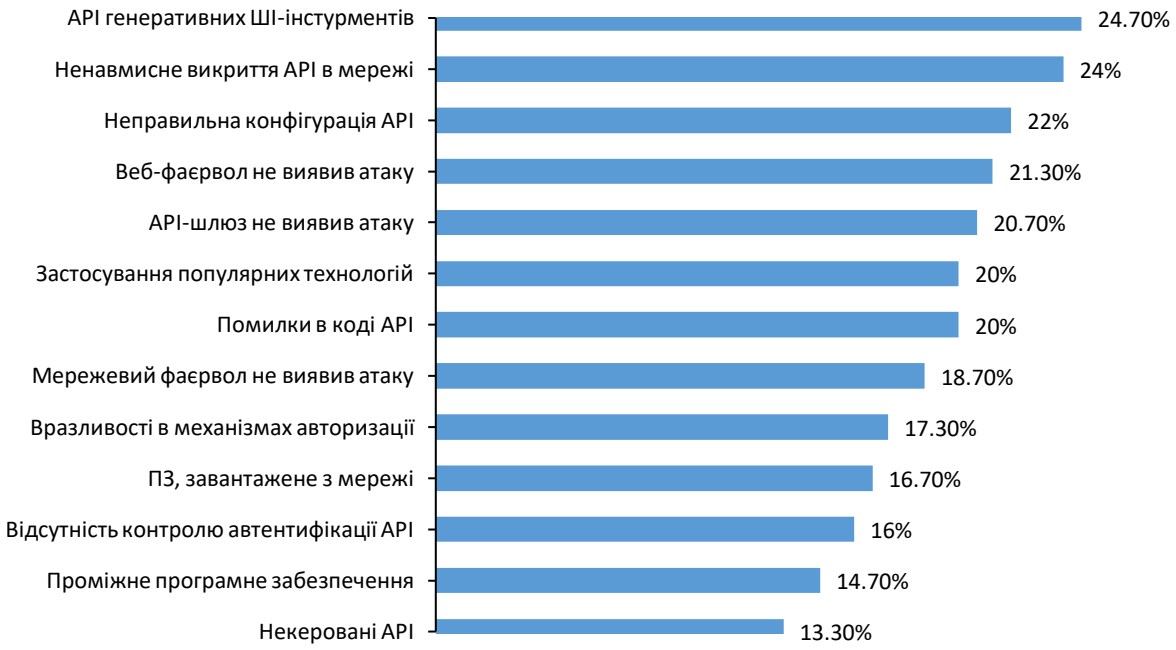

Рисунок 1.4 – Головні чинники виникнення інцидентів кібербезпеки API

Загалом, 44% усіх веб-атак проти комерційних організацій були спрямовані саме на API. Тому можна стверджувати, що обсяги, масштаби та рівень API атак

проти компаній електронної торгівлі зростають. Це включає використання зловмисниками ботів на основі генеративного ШІ, що дозволяє обходити традиційні засоби безпеки. Таким чином, все більше команд безпеки стикаються з даними загрозами та відчувають їх наслідки, як фінансові, так і людські.

1.2 Загальна характеристика Web API

Сучасні інформаційні системи і сервіси дедалі більше орієнтуються на архітектуру “клієнт-сервер”, де Web API виконує роль головного зв’язку між клієнтськими застосунками та серверною логікою. Оскільки веб-API надає програмний доступ до внутрішніх компонентів ІС і часто оперує конфіденційними даними користувачів, питання захисту таких інтерфейсів набуває особливої критичності. Тому для побудови і захисту прикладного програмного інтерфейсу, необхідно розуміти механізм роботи веб-API, з яких компонентів складається та яким чином клієнтські запити обробляються на сервері.

Web API визначає набір правил і форматів, за якими клієнт може звертатися до серверу, а саме [5]:

- Як передавати параметри та дані для авторизації.
- У якому форматі надавати та отримувати відповіді.
- Які HTTP-методи використовувати для виконання запитів.
- Як отримати доступ до певного ресурсу.
- Що означає кожен статус-код у відповідях.

У галузі електронної торгівлі такі прикладні програмні інтерфейси відіграють ключову роль не лише для внутрішньої логіки обробки замовлень та каталогів товарів, а й для швидкої і безпечної інтеграції з платіжними шлюзами, службами доставки, аналітичними й маркетинговими платформами, системами управління інвентарем, тощо. Завдяки використанню одного протоколу обміну даними та узгодженим форматам клієнт і сервер, а також усі зовнішні компоненти, можуть

працювати без помилок незалежно від мови програмування чи технологічної платформи.

1.2.1 Ключові компоненти веб-API

Web API складається з багатьох взаємопов'язаних частин, які забезпечують дотримання встановлених правил комунікації систем, і разом з тим, гарантують передбачувану поведінку сервісів, ефективність та швидкість обробки інформації. Сьогодні у переважній більшості прикладних програмних інтерфейсів можна помітити наступні компоненти [6, 7]:

- API-клієнт.
- API-сервер.

API-клієнт. Клієнтом прикладного програмного інтерфейсу є будь-який споживач сервісу, який формує запити до веб-API і обробляє отримані результати свого запиту. Зазвичай це може бути веб-додаток у браузері, мобільний застосунок, скрипт чи зовнішня програма. Завданням клієнтам є підготовка необхідних даних для формування запиту та обробка відповіді серверу.

Запит до серверу Web API об'єднує у собі інформацію, необхідну для коректного виконання завдання сервером та надання потрібної відповіді клієнту.

Точки доступу або кінцеві точки – це певні URI-адреси, за якими розміщений певний ресурс сервера, та за якою клієнт звертається для отримання або зміни інформації.

Для обмеження доступу до певних ресурсів використовуються методи автентифікації. Щоб отримати доступ до захищеного ресурсу, клієнт повинен у запиті передати свої облікові дані для підтвердження легітимності доступу. Найпоширенішими підходами на сьогодні є bearer-токени, OAuth 2.0 та API-ключі.

Кожен API-запит повинен містити метод, який визначає операцію, яку клієнт хоче здійснити на заданому ресурсі. Стандартними HTTP-методами є “GET”, “POST”, “PUT”, “PATCH” та “DELETE”, які позначають такі основні дії як отримання, створення, оновлення або видалення даних.

Параметри запиту – це певні змінні, які передаються кінцевій точці API для надання конкретних інструкцій обробки даних сервером. Зазвичай, ці інструкції включаються в запит як частина URL-адреси у рядку запиту, але їх також можна передавати у тілі запиту.

Заголовки запитів є парами “ключ-значення”, які надають додаткову інформацію про запит. Наприклад, заголовок “Content-Type” визначає формат даних у тілі запиту, а “Authorization” – облікові дані для автентифікації, такі як API-ключ або OAuth-токен.

Тіло запиту містить фактичні дані, необхідні для створення, оновлення або видалення ресурсу. Наприклад, якщо адміністратору магазину електронної комерції необхідно додати новий продукт, то тіло запиту може містити ім’я продукту, виробника, ціну, тощо. Специфікація конкретного веб-API буде вказувати на формат даних, необхідний для здійснення запиту, найпопулярнішими на сьогодні є формати JSON та XML.

API-сервер. Після того, як API-клієнт сформував запит, він надсилає його на відповідну точку доступу на сервері Web API для обробки. Далі сервер відповідає за обробку автентифікації, перевірку вхідних даних, отримання або модифікації даних з бази даних та повернення належної відповіді до клієнта.

Коли обробку інформації, наданої клієнтом, завершено, сервер формує відповідь, яка зазвичай містить код статусу, заголовок відповіді та тіло відповіді.

Код статусу API відповідають HTTP кодам статусу, які повертаються, щоб надати клієнту інформацію про результати запиту та допомогти йому зрозуміти, як діяти далі. Найчастіше зустрічають коди, які позначають успіх (2xx), помилку на стороні клієнту (4xx) або помилку на стороні серверу (5xx).

Заголовки відповідей HTTP є дуже схожими на заголовки запитів, за винятком того, що вони надають додаткову інформацію щодо відповіді сервера. Наприклад, заголовок “Cache-Control” вказує, як довго дані будуть зберігатися в кеші, а “Set-Cookie” – встановлює файл cookie для браузера, який може використовуватися для керування сеансом або автентифікації.

Тіло відповіді містить дані, які сервер повертає у відповідь на запит клієнта. Відповіді можуть досить сильно відрізнятися, але зазвичай вони включають об’єкти структурованих даних, які представляють запитувані ресурси, метадані або повідомлення про помилку. Наприклад, для успішного “GET” запиту певного продукту сервер може повернути JSON представлення даного продукту разом з міткою часу та джерелом даних.

Ще одним важливим, хоч і не обов’язковим, елементом прикладного програмного інтерфейсу є API-шлюз [8]. Він виступає як проміжний рівень між клієнтом і сервером, забезпечуючи централізовану маршрутизацію, балансування навантаження, кешування та контроль доступу. Шлюз приймає усі вхідні HTTP-запити, перевіряє їхню автентифікацію й авторизацію, проводить попереднє кешування відповідей для зменшення навантаження на мікросервіси.

Повна документація веб-API є важливим елементом будь-якого прикладного програмного інтерфейсу, оскільки вона допомагає стороннім розробникам швидко розібратися в роботі Web API, зрозуміти правила побудови запитів, форматі відповідей та коректності обробки помилок. Зазвичай документація містить опис усіх доступних кінцевих точок, методів, параметрів, форматів даних, приклади запитів та відповідей та правила автентифікації та авторизації.

1.2.2 Класифікація API у веб застосунках

Як вже було згадано раніше, Web API виконують велику кількість різнопланових задач і для задоволення потреб кожної окремої організації зазвичай створюються унікальні прикладні програмні інтерфейси. Таким чином, існує безліч

API, які значно відрізняються один від одного відповідно до поставлених цілей, технічних вимог та бізнес-моделей.

З огляду на таку різноманітність, без чіткої класифікації веб-API дуже складно забезпечити їх ефективний захист. Розуміння характерних рис і особливостей власно прикладного програмного інтерфейсу є досить важливим елементом його безпеки, оскільки воно дає змогу визначити політики безпеки й обмеження доступу, які відповідають конкретному типу інтерфейсу.

На рисунку 1.5 зображено класифікацію API у контексті кібербезпеки. Було виділено п’ять основних критеріїв, за якими можна охарактеризувати будь-яких інтерфейс, визначити потенційні вектори атак і загрози, та відповідно до цього встановити необхідні політики безпеки та обмеження доступу:

- За архітектурою.
- За рівнем доступу.
- За бізнес-моделлю.
- За життєвим циклом.
- За областю мережевого трафіку.

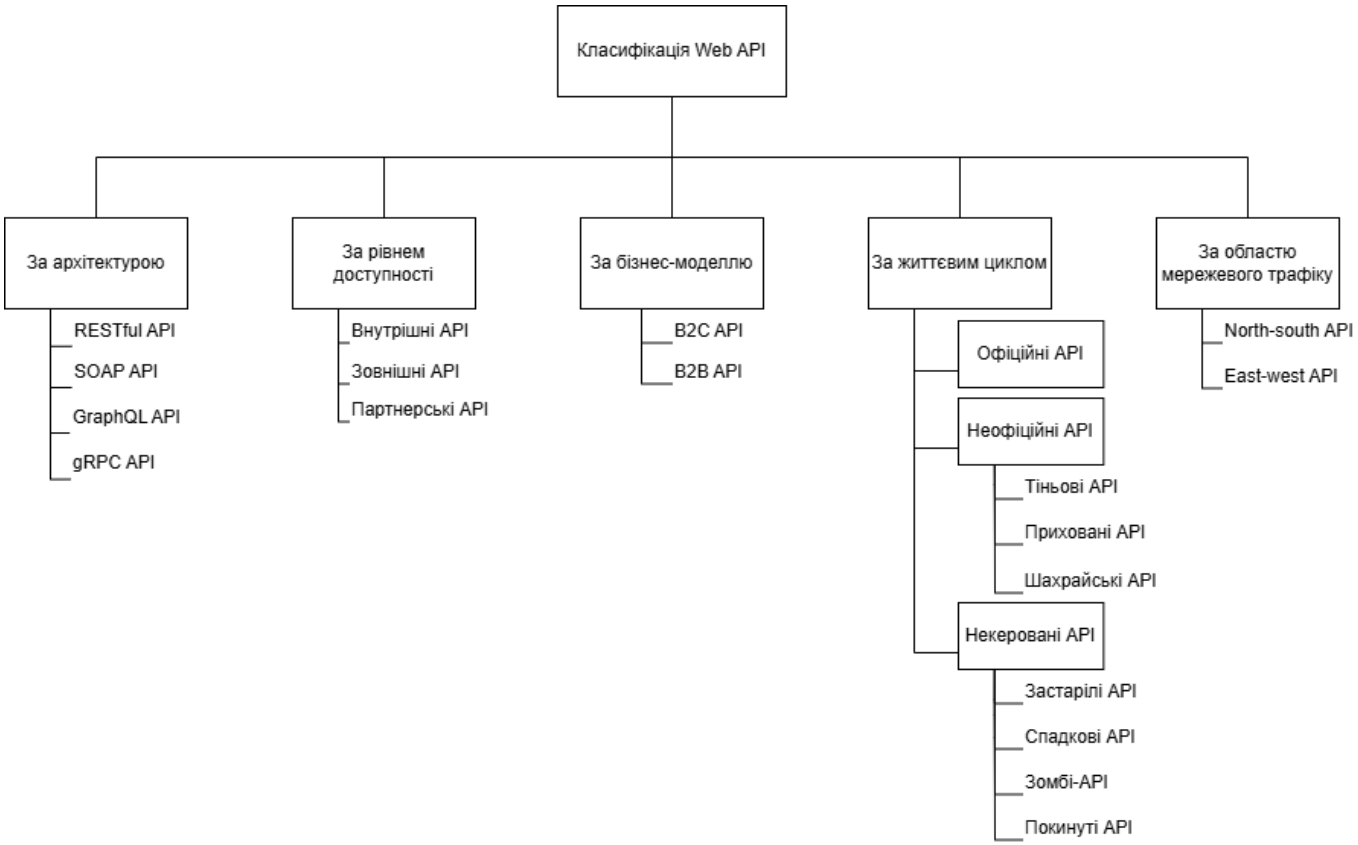

Рисунок 1.5 – Класифікація Web API

Архітектурний стиль API безпосередньо визначає, як клієнт і сервер обмінюються даними, які протоколи та формати використовують, а також як організовано самі виклики прикладного програмного інтерфейсу. Класифікація веб-API за даною категорією дозволяє сформувати чіткий список типових вразливостей, притаманних конкретному стилю. На сьогодні існує чотири найбільш популярні архітектури побудови Web API [9]:

- Restful API.
- SOAP API.
- GraphQL API.
- gRPC API.

Ще однією важливою класифікацією інтерфейсів є за рівнем доступності, оскільки для забезпечення належного рівня інформаційної безпеки необхідно розуміти, звідки саме можна отримати доступ до ресурсів. Серед даної категорії найчастіше виділяють внутрішні, зовнішні та партнерські API [10]. Внутрішні веб-

API призначені виключно для використання всередині організації, дозволяючи різним відділам або підрозділам ефективно обмінювати інформацією і використовувати внутрішні сервіси, та передбачають жорстіші правила мережевого доступу, у той же час, зовнішні – доступні будь-яким зовнішнім користувачам або стороннім розробникам та зазвичай мають відкриту документацію. Доступ до партнерських інтерфейсів надається за попередньою домовленістю з конкретними партнерами і часто здійснюється за допомогою окремих облікових записів, сертифікатів чи токенів.

Розрізнення веб-API за бізнес-моделлю потрібне, оскільки дає змогу зрозуміти, хто саме може здійснювати доступ та виконувати запити. У даній категорії виділяють B2C та B2B інтерфейси [11]. B2C (Business-to-consumer) API зазвичай забезпечують роботу мобільних та веб-застосунків, надаючи користувачам доступ до функціональних можливостей, а B2B (Business-to-Business) надає інтерфейс для використання іншими організаціями, інколи для надання послуг їм, а інколи – спільним користувачам.

Визначення життєвого циклу прикладного програмного інтерфейсу є важливим етапом для забезпечення належного рівня його безпеки. За даною категорією виділяють три основні групи веб-API: офіційні, неофіційні та некеровані інтерфейси [11]. До офіційних API відносять інтерфейси, які пройшли формальне затвердження, мають повну документацію, над якими проводяться регулярні тестування, випускаються оновлення та виконуються виправлення вразливостей. У свою чергу, неофіційні прикладні програмні інтерфейси зазвичай з’являються на етапі розробки та не проходять офіційно затвердження в організації, їх часто вводять для перевірки виконання певних бізнес-функцій компанії. Серед них найчастіше виділяють тіньові, шахрайські та приховані веб-API. Останньою великою групою інтерфейсів є некеровані прикладні програмні інтерфейси. До цього типу відносять Web API, які продовжують своє існування в системі, але з певних причин вже не підтримуються і вважаються застарілими. Прикладами таких інтерфейсів є застарілі, спадкові, “зомбі-API” та покинуті API.

Останньою, і не менш важливою, є класифікація інтерфейсів за областю мережевого трафіку, оскільки чітко визначивши межі, в яких відбувається передача даних, можна звузити та охарактеризувати перелік ймовірних джерел загроз. У цій категорії виділяють два типи веб-API: North-South та East-West API [11, 12]. До North-South відносять інтерфейси, якщо їх трафік має напрямок до або з контрольованої зони організації, а East-West – якщо передача мережевого трафіку відбувається виключно всередині контрольованої зони компанії.

1.2.3 Загрози безпеки для прикладних програмних інтерфейсів

Як вже згадувалося у попередніх розділах, надійні заходи безпеки API є критично важливими для організацій у галузі електронної торгівлі, особливо враховуючи конфіденційний характер даних, які проходять через інтерфейси.

Ігнорування проблем безпеки Web API може призвести до значних і незворотніх наслідків, зокрема для систем електронної комерції. Наприклад, недостатня увага до захисту прикладного програмного інтерфейсу може стати причиною витоку конфіденційних даних користувачів, що призведе до значних штрафів від регуляторів, репутаційної шкоди та втраті довіри користувачів.

Значної складності додає розглянута різноманітність API, однак, незважаючи на значні особливості кожної архітектури та правил комунікації, існують типові ризики інформаційній безпеці прикладних програмних інтерфейсів, які формують постійний ландшафт загроз [13, 14], відображений у таблиці 1.1.

Таблиця 1.1 – Поширені загрози безпеки API

Загроза	**Причини**	**Наслідки**
1	**2**	**3**
Ін’єкційні атаки	Відсутність або недостатня перевірка та очищення вхідних даних у запитах до API	Несанкціонований доступ до бази даних, витік або зміна даних, компрометація системи

Продовження таблиці 1.1

1	2	3
Неправильна автентифікація та управління сесіями	Слабкі механізми автентифікації, відсутність надійного завершення сесій, незахищені облікові дані	Викрадення сесій, несанкціонований доступ до функцій API, компрометація облікових записів користувачів
Небезпечна передача даних	Використання застарілих або вразливих протоколів, використання протоколів без наявності шифрування	Атаки “людина посередині”, перехоплення або підміна даних, порушення конфіденційності та цілісності переданих повідомлень
Атаки відмови в обслуговуванні	Відсутність обмежень на кількість запитів, недостатня масштабованість інфраструктури	Перевантаження сервера, недоступність сервісів API, порушення роботи бізнес-процесів
Неправомірне використання ключів API	Неналежне зберігання ключів, відсутність регулярної ротації ключів, слабкий контроль доступу	Несанкціонований доступ до функцій API, крадіжка або підробка даних, компрометація сервісу
Недостатня перевірка вхідних даних	Відсутність перевірки формату, типу або значень параметрів запиту	Помилки або збої у роботі API, можливість ін’єкцій
Неконтрольовані перенаправлення та переспрямування	Відсутність перевірки цільових URL-параметрів у функціях перенаправлення	Викрадення облікових даних, компрометація безпеки клієнтів або партнерів
Неконтрольовані перенаправлення вхідних даних	Відсутність фільтрації або перевірки переданих параметрів перед виконанням операцій перенаправлення	Компрометація внутрішніх компонентів API, відправлення шкідливих даних до сервера

Кінець таблиці 1.1

1	2	3
Відсутність контролю доступу	Невірно налаштовані права доступу, відсутність перевірок авторизації перед обробкою запитів, недосконала система ролей	Несанкціонований доступ до конфіденційних ресурсів, витік або модифікація даних, порушення цілісності системи
Відсутність моніторингу та логування	Відсутність налаштувань журналювання запитів та відповідей, ігнорування необхідності відстеження аномалій у трафіку API	Затримка реагування на інциденти, збільшення масштабу загрози, ускладнення розслідування

1.3 Інструменти тестування безпеки на основі аналізу Web API

Зважаючи на значну кількість безпекових загроз, тестування безпеки API стає важливою практикою для компаній електронної комерції. Воно передбачає систематичний процес виявлення вразливостей прикладного програмного інтерфейсу до того, як ними зможуть скористуватися зловмисники. Впровадження таких інструментів у політику безпеки організації має ряд переваг, які мають значний відбиток на успішності системи електронної торгівлі та її безпеці [15].

Перш за все, проведення тестувань безпеки забезпечує проактивний підхід шляхом розвідки та дослідження власного середовища, що дає організації можливість превентивної дії, яка мінімізує шанси виникнення певного інциденту кібербезпеки. Регулярне тестування безпеки API допомагає досягти та підтримувати відповідність галузевим стандартам безпеки та нормам, зокрема ISO/IEC 27005 [16, 17], що може вберегти компанію від значних штрафів та репутаційних втрат. Також важливим фактором є довіра з боку користувачів, оскільки надаючи перевагу безпеці веб-API організація демонструє відповідальність за безпеку клієнтських даних. Це є життєво важливим елементом будь-якої успішної системи електронної торгівлі, тому

застосування інструментів для тестування кібербезпеки власних інтерфейсів сьогодні є необхідністю.

Існують різні методи проведення тестувань безпеки зі своїми перевагами та недоліками, однак, зазвичай виділяють чотири основі підходи [18]:

- Статичний аналіз – статичні тести безпеки API виконуються для перевірки вихідного коду шляхом пошуку небезпечних шаблонів розробки.
- Динамічний аналіз – динамічні тести перевіряють вже запущені прикладні програмні інтерфейси на наявність вразливостей та помилок, які можна використати. Даний підхід не потребує доступу до програмного коду веб-API, оскільки він використовує підхід “ззовні всередину”, симулюючи атаки зловмисника.
- Фаз-тестування – за даного типу тестування автоматично створюються та вводяться випадкові, недійсні або інші неочікувані вхідні дані, щоб перевірити, як API буде на них реагувати та чи будуть виникати помилки у роботі.
- Тестування на проникнення – ці тестування проводяться шляхом імітації атак на Web API з внутрішніх чи зовнішніх джерел, намагаючись використати вразливості, команді безпеки вдається ідентифікувати та запобігти потенційним вразливостям інтерфейсу.

1.3.1 Огляд популярних інстурментів тестування кібербезпеки API

Існує велика кількість різних інструментів тестування безпеки на основі аналізу Web API, які відрізняються один від одного за значною кількістю параметрів, наприклад, такі як ціна або наявні типи тестування. Однак варто розглянути найпопулярніші на сьогодні інструменти [15], а саме: 42Crunch, Jit, Salt Security, Pynt, Postman & SoapUI.

42Crunch. 42Crunch – це SaaS-платформа для безпеки API, яка забезпечує контрактно-орієнтований підхід [19]. Основна ідея платформи полягає в тому, що весь цикл безпеки базується на конкретному OpenAPI/Swagger-контракті, на основі

якого виконується статичний аудит. Під час аудиту платформа аналізує контракт за певним набором правил і формує оцінку захищеності, яка відображає рівень ризику API, а також надає рекомендації з виправлення виявлених недоліків.

На основі контракту 42Crunch може виконувати динамічне сканування живого прикладного програмного інтерфейсу шляхом генерації HTTP-запитів, з характерним для типових атак корисним навантаженням, та порівняння відповідей з тими, що передбачені контрактом. Якщо реалізація веб-API повертає некоректні дані, ігнорує авторизацію чи порушує схему відповіді, платформа відразу повідомляє про невідповідність і формує детальний звіт із відтворюваними кроками для розробників.

Важливою складовою 42Crunch є модуль API Protect, який фактично виконує роль мікрофаєрвола. На основі OpenAPI-файлу платформа автоматично генерує набір правил, які визначають дозволені HTTP-методи, шляхи, параметри та формати вхідних запитів і відповідей. Крім того, даний модуль має можливість контролю автентифікації та авторизації, накладання обмежень на кількість запитів та захисту від класичних injection-атак.

Інтеграція 42Crunch у CI/CD здійснюється через готові плагіни для популярних систем, таких як GitLab CI, GitHub Actions, Jenkins, Azure DevOps, Bamboo, Bitbucket Pipeline. Під час запуску, плагін шукає у репозиторії всі OpenAPI-файли, завантажує їх на платформу і виконує статичний аудит безпеки. Якщо хоча б один контракт не відповідає мінімальним пороговим значенням для кількості критичних, високих чи середніх вразливостей, то така збірка далі не проходить.

Окрім базових модулів, у веб-консолі 42Crunch всі імпортовані OpenAPI-файли завантажуються у колекції, що дає змогу групувати API за командами, проєктами чи середовищами та налаштовувати доступ за ролями. Платформа також підтримує автоматичне виявлення некерованих Web API та функції моніторингу та аналізу, що дозволяє отримати статистику за найбільш атакованими шляхами, способами атак тощо. Вигляд панелі керування 42Crunch показано на рисунку 1.6.

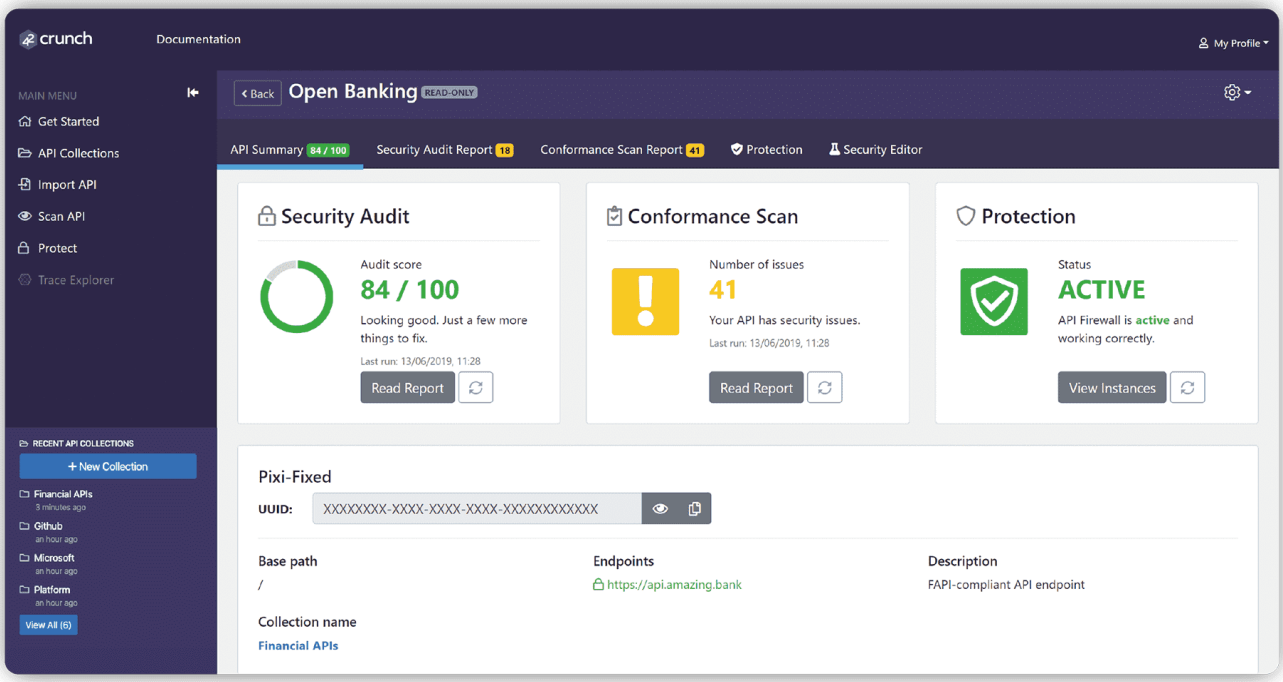

Рисунок 1.6 – Панель керування 42Crunch [19]

Jit. Jit – це комплексна платформа, яка об’єднує статичний, динамічний та компонентний аналіз під єдиним інтерфейсом [20]. Головна ідея полягає у тому, що кожен етап життєвого циклу інтерфейсу супроводжується автоматичними перевірками безпеки, а результати зберігаються у єдиному середовищі, яке допомагає пріоритизувати й фільтрувати вразливості.

На рівні статичного аналізу Jit виконує аналіз вихідного коду і виявляє класичні вразливості. Важливою особливістю платформи є формування переліку усіх використаних компонентів, наприклад, бібліотек або сторонніх залежностей, та проведення аналізу складу програмного забезпечення, перевіряючи компоненти за базою даних вразливостей.

Для виконання динамічних сканувань платформа використовує інтеграцію з OWASP ZAP. Вона дозволяє автоматично генерувати запити, які імітують популярні атаки, перевіряти бізнес-логіку API, обмеження на кількість запитів та стійкість до атак грубої сили. Особливістю є можливість створення власних правил безпеки для специфічних загроз, а також запуск динамічних сканувань як з хмарного середовища,

так і локально, що дозволяє проводити тестування внутрішніх прикладних програмних інтерфейсів.

Інтеграція Jit у CI/CD реалізована через агенти інтерфейсів командного рядка та готові плагіни для платформ GitHub Actions, GitLab CI, Jenkins, Azure DevOps. При кожній спробі додавання коду до репозиторію автоматично виконується статичний аналіз коду, а перед розгортанням у тестове середовище – динамічне сканування.

Для звітування та аналізу платформа пропонує наступні засоби:

- Інформаційні панелі з візуалізацією загроз, трендами вразливостей та розподілом за мікросервісами чи командами.
- Інтерфейс моніторингу та керування вразливостями (рисунок 1.7), який автоматично пріоритизує вразливості відповідно до середовища та ризиків.
- Звіти безпеки, які можуть бути експортовані у CSV, JSON або PDF форматах. Jit також підтримує автоматичне створення завдання в Jira для критичних проблем з переліком необхідних дій для швидкого реагування.

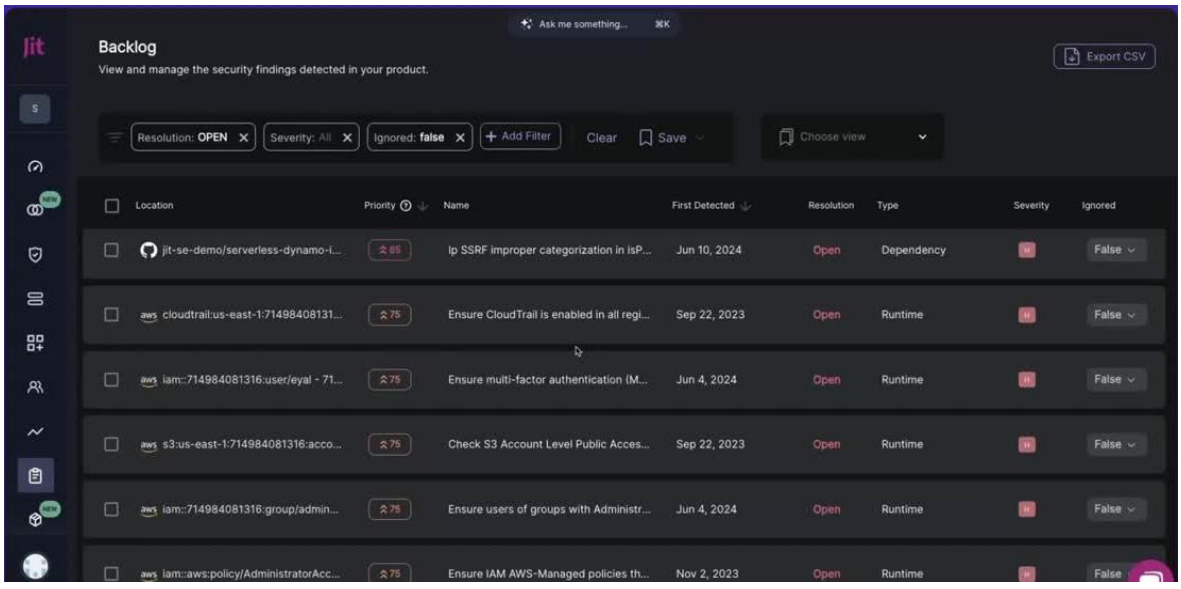

Рисунок 1.7 – Інтерфейс моніторингу та керування вразливостями Jit [20]

Salt Security. Salt Security – це хмарна платформа захисту API у реальному часу, яка передусім орієнтована на виявлення загроз під час виконання [21]. На відміну від традиційних рішень з статичним або динамічним аналізом, Salt Security не виконує активне сканування прикладного програмного інтерфейсу перед розгортанням, натомість платформа аналізує реальний трафік API, застосовуючи AI/ML-моделі для

виявлення аномалій та потенційних атак. Після початкового розгортання Salt підключається до API-шлюзу або сервера через механізми віддзеркалення трафіку, щоб доступ до отриманих запитів.

Основною складовою платформи є модуль виявлення загроз, яка вивчає запити до веб-API, щоб побудувати шаблон нормальної поведінки. Використовуючи алгоритми машинного навчання, Salt Security автоматично виявляє нетипові запити та адаптується до змін. У разі підозрілих дій платформа формує інцидент і, залежно від налаштувань, може автоматично блокувати такі виклики.

Платформа також пропонує функції виявлення некерованих API, вона безперервно відстежує кінцеві точки Web API, які незадокументовані в OpenAPI/Swagger-контрактах, і формує поточний інвентар прикладних програмних інтерфейсів. Це дозволяє виявляти раніше невідомі вектори атак. Крім того, Salt може виконувати автоматичну оцінку політик безпеки існуючого контракту, повідомляючи про наявні вразливості.

Salt Security надає деталізовані звіти, які включають інформацію про тренди атак, статистику підозрілої поведінки, оцінку ризику кожної кінцевої точки з рекомендаціями по їх зменшенню. Дані звіти можна експортувати у CSV або PDF форматах та налаштувати автоматичне сповіщення. Панель керування Salt Security зображена на рисунку 1.8.

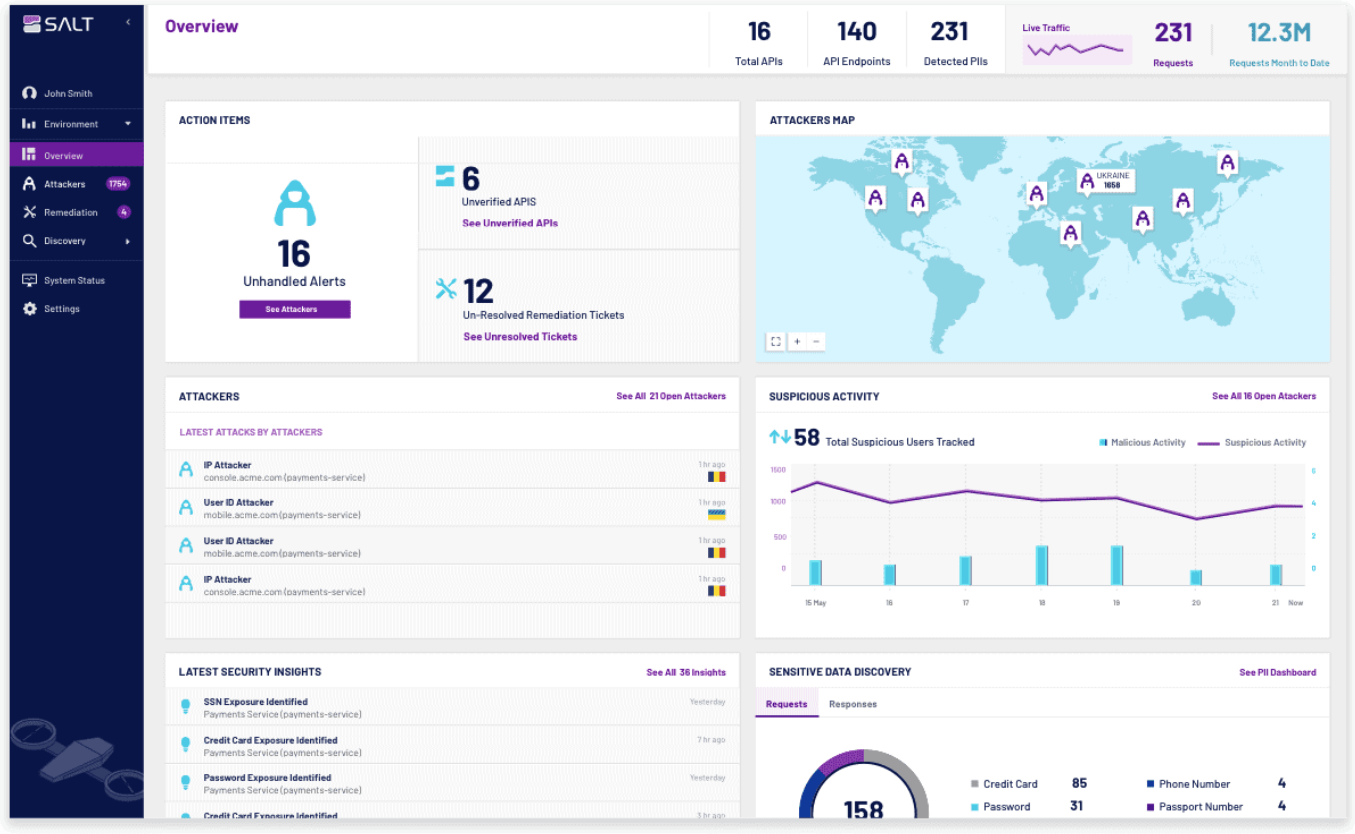

Рисунок 1.8 – Панель керування Salt Security [21]

Pynt. Pynt – це CLI-орієнтований інструмент для динамічного аналізу і перевірки безпеки API перед розгортанням [22]. На відміну від традиційних рішень з графічним інтерфейсом користувача, Pynt встановлюється в програмному середовищі розробника або інтегрується в CI/CD. Інструмент автоматично аналізує всі наявні функціональні тести, генерує звичайні та шкідливі HTTP-запити, а також аналізує результати відповідей, щоб визначити потенційні вразливості.

Характерною особливістю Pynt є те, що інструмент вивчає відповіді API та на їх основі формує сценарії, які імітують як звичайний, так і шкідливий трафік. Наприклад, якщо функціональні тести відправляють запити із певним набором полів у тілі, то інструмент автоматично додає у них шкідливе навантаження для поширених типів атак, враховуючи структуру очікуваної відповіді.

Окрім цього, інструмент також пропонує модуль виявлення некерованих API, який сканує середовище під час виконання тестів і виявляє незадокументовані в OpenAPI-контрактах прикладні програмні інтерфейси. На основі цих даних Pynt формує поточний інвентар веб-API, що допомагає знайти невідповідності в контракті.

Pynt легко інтегрується у CI/CD за рахунок того, що достатньо додати один рядок у скрипт збірки, щоб після виконання функціональних тестів відразу почати перевірку безпеки за допомогою Pynt. Це дозволяє інтегрувати інструмент з майже будь-яким CI/CD рішенням на ринку. Також Pynt підтримує інтеграцію з Postman та ReadyAPI через відповідні конфігурації CLI, що дозволяє не переписувати наявні тести. Інструмент дозволяє генерувати CLI-звіти у JSON та CSV форматах із деталями кожного інциденту. Панель керування Pynt показано на рисунку 1.9.

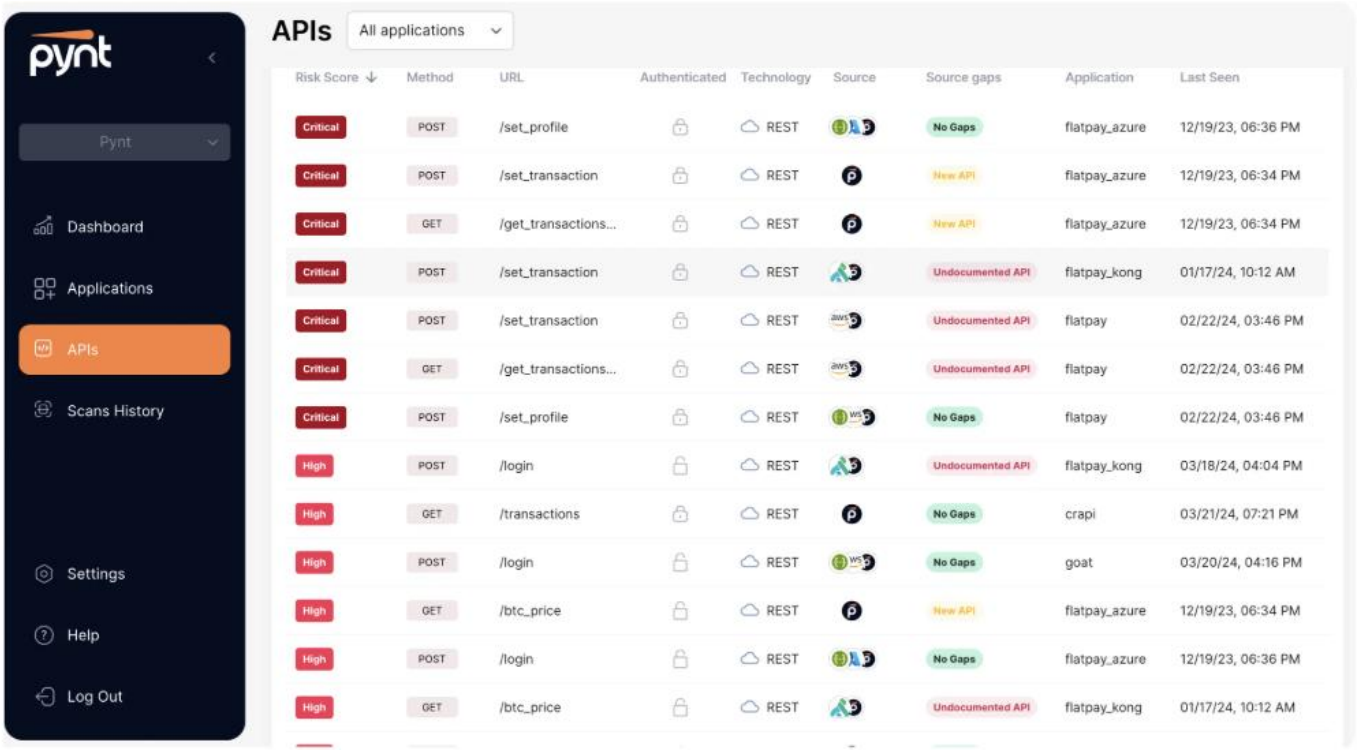

Рисунок 1.9 – Панель керування Pynt [22]

Postman & SoapUI. Postman & SoapUI – це універсальні платформи для тестування API, які також пропонують базові можливості для тестувань безпеки [23, 24]. Хоча їхній основний функціонал зосереджений на функціональному та навантажувальному тестуванні, обидва інструменти мають вбудовані або розширювані модулі для перевірки поширених вразливостей.

Postman дозволяє легко створювати та зберігати набори запитів, визначати змінні середовища і писати прості перевірки у вигляді скриптів JavaScript, що дозволяє швидко виявляти помилки в логіці обробки запитів. Наявна також

автоматична перевірка контракту API для порівняння документованої поведінки сервісу з реальною.

Для забезпечення базового рівня безпеки Postman дозволяє вбудовувати в колекції запитів тести, які перевіряють, наскільки коректно сервер обробляє вхідні дані. Крім того, у ньому є вбудована функція пошуку випадково залишених у коді API-ключів та паролів, що запобігає витоку даної інформації.

Для інтеграції у CI/CD Postman використовує CLI-інтерфейс Newman, який дає змогу запускати колекції тестів безпосередньо після кожної модифікації коду. Наявні готові шаблони для інтеграції у популярні рішення безперервної інтеграції та доставки, а після виконання Newman формує звіти у форматах HTML, JSON або XML, де вказуються як успішні, так і провалені тести. Панель керування Postman зображено на рисунку 1.10.

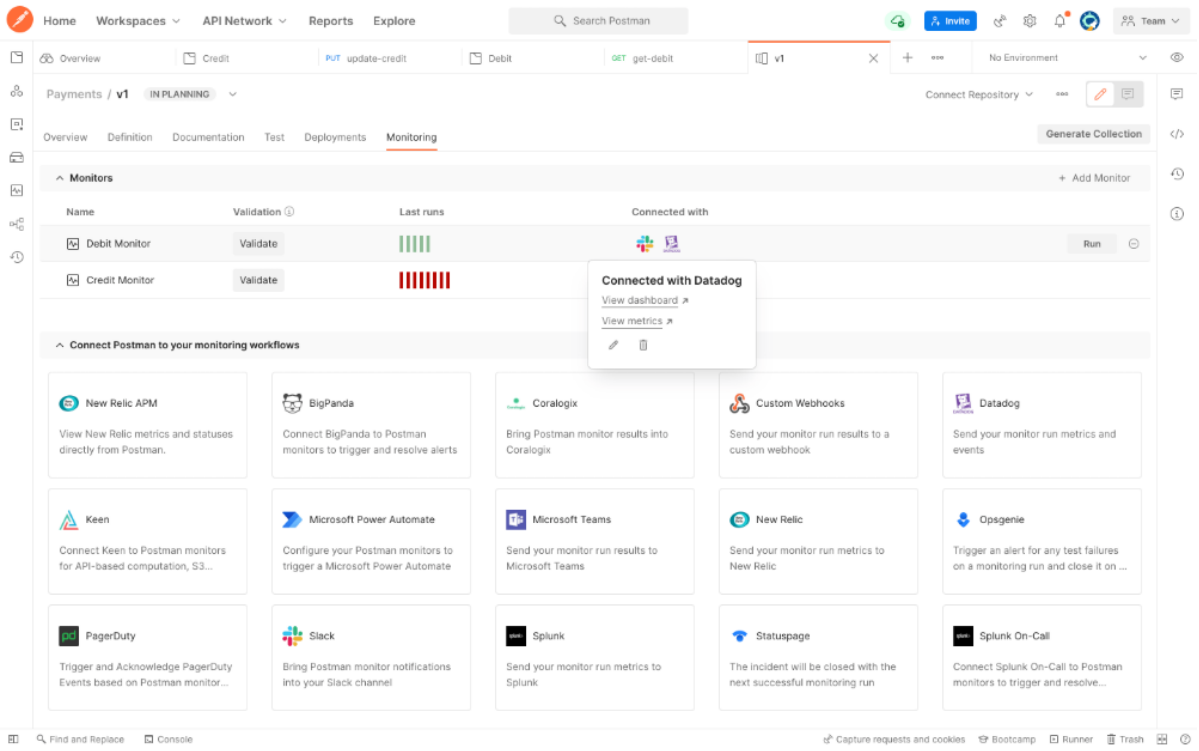

Рисунок 1.10 – Панель керування Postman [23]

SoapUI, наразі більш відомий як ReadyAPI, зосереджується передусім на глибокому тестуванні інтерфейсів. Перш за все, він забезпечує функціональне тестування інтерфейсів із можливістю створювати складні запити, перевіряти відповіді за схемами і сценаріями, а також керувати тестовими даними через параметризацію.

Він забезпечує тестування безпеки прикладного програмного інтерфейсу поєднуючи автоматичні OWASP-орієнтовані перевірки та фаз-тестування. Після імпорту контракту API платформа генерує набір кінцевих точок і автоматично створює шкідливі навантаження для запитів. Завдяки цьому, ReadyAPI може виявляти такі вразливості, як неправильна перевірка вхідних значень, обходи авторизації та порушення обмежень на кількість запитів.

Наприкінці кожного циклу тестування ReadyAPI формує зведений звіт з усіма виявленими проблемами, розподілом за категоріями та рекомендаціями щодо їх вирішення. Існує можливість автоматизації через командний рядок та CI/CD плагіни.

Інтеграція ReadyAPI з Postman реалізується через можливість імпорту та експорту колекцій, що дозволяє імпортувати Postman набори тестів разом із середовищами та змінними, конвертуючи їх для використання у ReadyAPI. Панель керування ReadyAPI показано на рисунку 1.11.

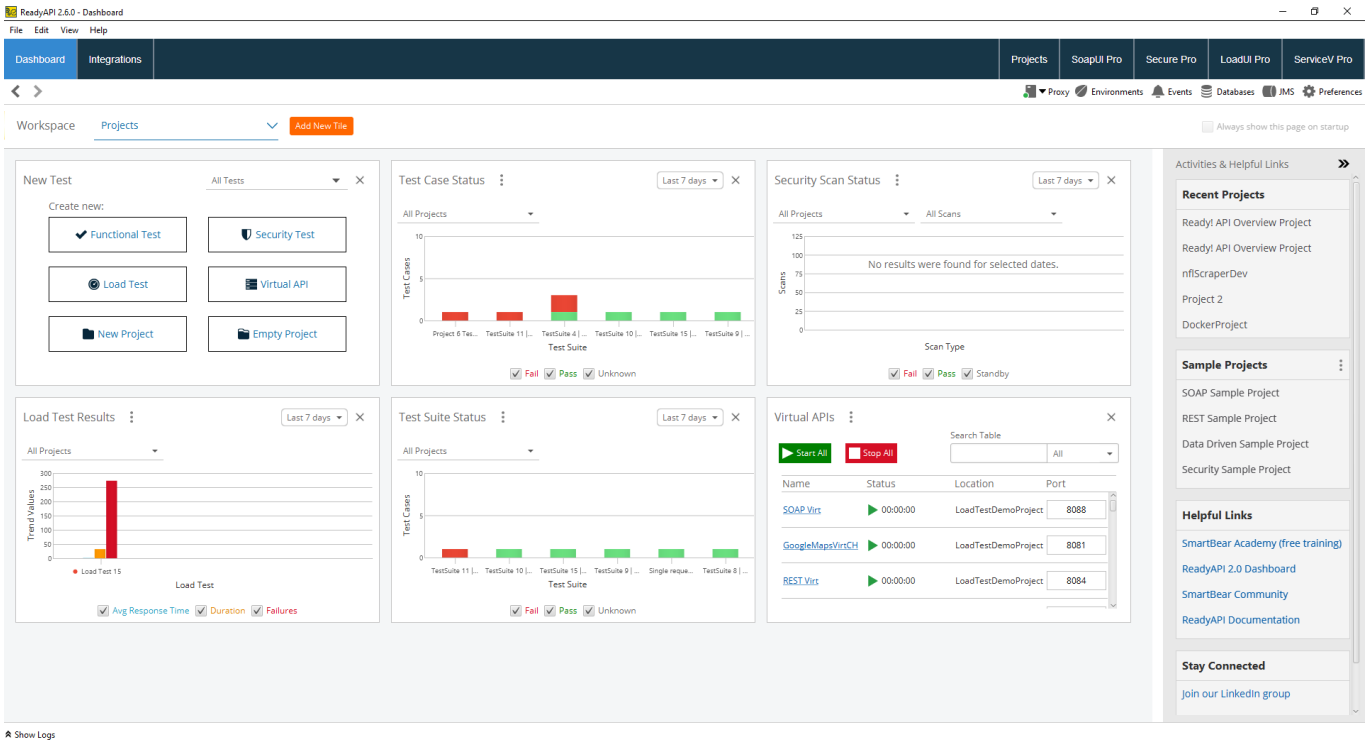

Рисунок 1.11 – Панель керування ReadyAPI [24]

1.3.2 Проблема вибору інструменту тестування безпеки веб-API

Зазвичай, будь-яка компанія бажає використовувати найкращий з існуючих на ринку інструментів тестування безпеки, але часто вона має певну обмежену кількість наявних ресурсів, що робить вибір засобу безпеки не таким однозначним.

З огляду інструментів тестування безпеки API стає зрозуміло, що всі вони значно відрізняються один від одного як з точки зору набору функціональних можливостей, так і з точки зору зручності та легкості застосування. Таким чином, організації, які вирішують впровадити використання подібного засобу, стикаються ще й з проблемою вибору оптимального інструменту тестування безпеки Web API серед безлічі доступних рішень.

З'являється потреба у порівнянні та оцінці групи відібраних інструментів за певними критеріями для об'єктивного вибору. З аналізу джерел [25, 26] стає зрозуміло, що єдиного підходу до вибору критеріїв оцінки не існує, наприклад, у першій статті для вибору оптимального інструменту виділяють такі критерії, як тип сканування, легкість використання, покриття вразливостей та звітування з усуненням вразливостей. У другій статті виділяють дещо інші критерії, на які варто звернути увагу: легкість використання, інтеграція з CI/CD, масштабування, підтримка багатьох архітектур, аналіз та звітування у реальному часі.

Після вибору критеріїв перед організацією постає ще одна проблема, а саме вибір релевантної методики оцінки. Використання громіздких методик у невеликих компаніях може бути непродуктивним через часові витрати на проведення оцінки, наприклад, збір даних та навчання працівників. З іншого боку, у великих корпораціях, де усі рішення мають бути обґрунтовані перед керівництвом та аудиторами, використання занадто простих методів може не дати належної об'єктивності в результатах проведеної оцінки.

Таким чином, проблема вибору інструменту тестування безпеки веб-API зводиться до двох основних аспектів: формулювання набору критеріїв, виходячи з можливостей та потреб організації, та вибору методики, яка дозволить провести об'єктивне оцінювання альтернатив відповідно до цих критеріїв.

Висновки до розділу 1

Організації у сфері електронної торгівлі все частіше зустрічаються з атаками спрямованими на їхні Web API. Тому перед ними постає питання захисту та тестування власних прикладних програмних інтерфейсів з метою зменшення кількості вразливостей та їх наслідків для інформаційної безпеки системи.

У даному розділі було розглянуто основні особливості прикладних програмних інтерфейсів, їх роль у роботі інформаційних систем та загрози безпеки, які зустрічаються найчастіше. Було розглянуто ключові компоненти переважної більшості сучасних інтерфейсів та процес взаємодії клієнта та сервера за допомогою API. Окрім цього, було сформовано класифікацію Web API, яка є важливим елементом у забезпеченні належного рівня безпеки.

Сьогодні існує велика кількість інструментів для тестування кібербезпеки на основі аналізу веб-API. Тенденція до збільшення атак на прикладні програмні інтерфейси систем електронної торгівлі демонструє необхідність у використанні подібних засобів захисту. З огляду таких інструментів як 42Crunch, Jit, Salt Security, Pynt та Postman & SoapUI стає очевидною їх значна відмінність у функціональних можливостях та зручності використання. З цих причин, актуальності набуває проблема вибору оптимального інструменту тестування.

2 МЕТОДИ БАГАТОКРИТЕРІАЛЬНОГО ПРИЙНЯТТЯ РІШЕНЬ ТА ФОРМУВАННЯ КРИТЕРІЇВ ВИБОРУ

2.1 Багатокритеріальне прийняття рішень

Багатокритеріальне прийняття рішень (MCDM) – це сукупність методів та підходів, спрямованих на вибір оптимального рішення серед альтернатив, що оцінюються за кількома критеріями. Складність проблеми прийняття вибору настає через наявність більше ніж одного критерію вибору, оскільки більше не існує унікального оптимального вирішення проблеми, яке можна було б отримати без інформації про переваги альтернатив.

Незважаючи на те, що активний розвиток методів MCDM розпочався у 1960 роках, та за цей час було розроблену велику кількість методів, які суттєво відрізняються один від одного, усі ці підходи поділяють деякі спільні особливості [27].

Альтернативи – це певні варіанти рішень або можливі об’єкти вибору, що порівнюються між собою за визначеними критеріями. Зазвичай, під час вирішення задачі вибору, вони позначаються у наступній математичній формі:

$$A = \{A_i | i = 1,2, \dots, m\} \tag{2.1}$$

Критерії – це параметри або властивості, за якими оцінюються альтернативи. Критерії відображають важливі для прийняття рішення аспекти, наприклад, вартість, якість, надійність і т.д. Хоча за своєю природою альтернативи є однорідними елементами, але це не є обов’язковим у випадку критеріїв, які можуть мати різні одиниці виміру або суперечити один одному. Критерії позначаються наступним чином:

$$C = \{C_j | j = 1,2, \dots, n\} \tag{2.2}$$

Ваги критеріїв – це нормалізований набір числових коефіцієнтів, які призначаються кожному критерію на основі його важливості. Ваги дозволяють з

урахуванням пріоритетів користувача або зацікавлених сторін адекватно зіставити критерії між собою. Математичний запис вагових критеріїв у MCDM зображений у формулі 2.3.

$$W = \{w_j | j = 1,2, \dots, n\} \tag{2.3}$$

Матриця вибору – це таблиця, яка представляє ефективність кожної альтернативи за кожним критерієм. Дана матриця та вектор вагових коефіцієнтів є основними вхідними даними для методів багатокритеріального прийняття рішень. Матриця вибору, у загальному випадку, позначається наступним чином:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} \tag{2.4}$$

Далі буде розглянуто два популярні методи багатокритеріального прийняття рішень для оцінки та впорядкування альтернатив, а саме TOPSIS та ELECTRE.

2.2 Метод TOPSIS

TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) – це метод багатокритеріального прийняття рішень, розроблений Чін-Лай Хвангом та Юном у 1981 році, який базується на концепції, що обрана альтернатива має мати найкоротшу геометричну відстань до позитивного ідеального рішення та найдовшу геометричну відстань до негативного ідеального рішення. TOPSIS, як і інші компенсаційні методи, дозволяє компроміси між критеріями, коли поганий результат за одним критерієм може бути усунений хорошим результатом за іншим критерієм.

Метод TOPSIS має досить простий процес виконання, який полегшує його програмування та використання, через що він є досить популярним серед дослідників. Головними перевагами даного методу можна виділити [28]:

- Має досить простий процес обчислення, що робить можливим його використання у електронних таблицях.

- Кількість альтернатив та критеріїв не вливає на кількість кроків виконання методу.
- Значення шкали методу може враховувати одночасно як найкращі, так і найгірші альтернативи.

Однак не можна недооцінювати та ігнорувати недоліки методу TOPSIS:

- У даному методі зважування критеріїв є складним і неточним процесом.
- Можна зіткнутися з альтернативами, які одночасно близькі до позитивної та негативної ідеальних точок.

2.2.1 Процес виконання методу TOPSIS

Процес вибору найкращої альтернативи використовуючи підхід TOPSIS можна описати шістьма послідовними кроками [28].

Першим кроком є нормалізація матриці вибору. В кожній MCDM задачі існують різні критерії з різними одиницями виміру, і щоб ці критерії можна було порівнювати між собою необхідно їх привести до однієї розмірності. Це досягається за допомогою процесу нормалізації і, в загальному випадку, використовується формула 2.5.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \tag{2.5}$$

Наступним кроком є обчислення зваженої нормалізованої матриці. Метою даного етапу є накладання певної ступені важливості кожного критерію на матрицю вибору. За результатами даного кроку буде отримано зважену матрицю вибору *V*, де $v_{ij} = w_j r_{ij}$.

$$V = \begin{bmatrix} v_{11} & \cdots & v_{1m} \\ \vdots & \ddots & \vdots \\ v_{n1} & \cdots & v_{nm} \end{bmatrix} \tag{2.6}$$

Третім кроком є визначення позитивного ідеального A^* та негативного ідеального A^- рішень. Вони можуть бути визначені за допомогою формул 2.7 та 2.8.

$$A^* = \{v_1^*, v_2^*, \dots, v_n^*\} = \left\{\left(\max_j v_{ij} | i \in I'\right), \left(\min_j v_{ij} | i \in I''\right)\right\}, \tag{2.7}$$

$$A^- = \{v_1^-, v_2^-, \dots, v_n^-\} = \left\{\left(\min_j v_{ij} | i \in I'\right), \left(\max_j v_{ij} | i \in I''\right)\right\} \tag{2.8}$$

Дані рівняння означають, що A^* отримується зі збору найкращих результатів матриці V, а A^- навпаки – з найгірших результатів нормалізованої матриці.

Після визначення позитивного та негативного рішень, наступним кроком є обрахунок дистанції альтернатив до A^* та A^-. Існують різні способи виконання даного обчислення, але найпопулярнішим є використання методу Евклідової відстані. Для позитивного та негативного рішень вона обчислюється формулами 2.9 та 2.10.

$$D_i^* = \sqrt{\sum_{j=1}^n (v_{ij} - v_j^*)^2}, \tag{2.9}$$

$$D_i^- = \sqrt{\sum_{j=1}^n (v_{ij} - v_j^-)^2} \tag{2.10}$$

П’ятим кроком виконання методу TOPSIS є обчислення відносної близькості. Значення коефіцієнту відносної близькості варіюється в діапазоні 0 та 1 і обчислюється за допомогою наступної формули:

$$C_i^* = \frac{D_i^-}{D_i^* + D_i^-} \tag{2.11}$$

Фінальним кроком є ранжування альтернатив. Елементи впорядковуються від найкращих до найгірших, з найбільшим (найближче значення C_i^* до 1) та найменшим значенням C_i^* відповідно. Таким чином, альтернатива, яка знаходитиметься найвище у списку, і буде вирішенням задачі вибору.

2.3 Група методів ELECTRE

ELECTRE (Elimination Et Choix Traduisant la Realite) – це сімейство методів багатокритеріального прийняття рішень, започатковане Бернаром Руа у 1966 році. Ці методи базуються на порівнянні альтернатив шляхом врахування певних заданих критеріїв порівняння. Головною відмінністю ELECTRE від компенсаційних методів прийняття рішень є те, що вагові коефіцієнти є коефіцієнтами важливості певного критерію, а не є коефіцієнтами заміни. Таким чином, неможливо компенсувати погане значення критерію певної альтернативи іншим критерієм з хорошим значенням. Також варто зазначити, що методи ELECTRE активно застосовуються в наукових працях та при вирішенні прикладних задач [29].

ELECTRE вважається одним з найкращих методів прийняття рішень за багатьма критеріями, оскільки він має ряд переваг [30]:

- Включає можливість роботи як з якісними, так і кількісними оцінками критеріїв.
- Може мати справу з різнорідними шкалами за рахунок нормалізації.
- Ваги критеріїв не є коефіцієнтами заміщення.
- Враховує недосконалості даних та довільність за рахунок порогів узгодженості та неузгодженості.

Однак потрібно розуміти, що як і будь-який інший метод багатокритеріального прийняття рішень, ELECTRE також має свої недоліки:

- Використання необґрунтованих порогових значень може суттєво впливати на результати.
- Даний метод вимагає значних обчислювальних ресурсів.

Методи групи ELECTRE найчастіше використовуються для вирішення проблем вибору, ранжування та сортування. Для цього було розроблено методи, які хоч і відповідають концепції оригінального підходу, але дещо відрізняються один від одного. У межах родини методів ELECTRE виокремлюють кілька основних версій,

кожна з яких оптимізована під конкретний клас задач. ELECTRE I застосовується переважно для базового вибору найкращого варіанту шляхом виділення альтернатив, які не поступаються жодній іншій альтернативі. ELECTRE II використовує дещо модифіковані процедури побудови порогових значень та проміжного рейтингу альтернатив, що дозволяє отримати впорядкований ряд альтернатив. ELECTRE III фокусується на ранжуванні та використовує індекс достовірності для роботи з нечіткими або суперечливими даними. Для задач сортування було створено ELECTRE IV та ELECTRE TRI, який класифікує альтернативи за заздалегідь визначеними класами.

2.3.1 Формалізація методів сімейства ELECTRE

У даній групі методів розглядаються m альтернатив (які позначаються як $A_i: i = 1, \dots, m$), n критеріїв ($X_j: j = 1, \dots, n$) та n вагових коефіцієнтів ($w_j: j = 1, \dots, n$). Важливо зазначити, що сума вагових коефіцієнтів повинна дорівнювати 1 ($\sum_{j=1}^{n} w_j = 1$). Зазвичай даний набір зображується у вигляді матриці розмірності $m \times n$. Для вибору найкращої альтернативи методи ELECTRE використовують переважаючі відношення між кожною парою альтернатив, де краща альтернатива повинна відповідати двом наступним умовам:

- щонайменше бути настільки ж хорошою за більшістю критеріїв, ніж інша альтернатива.
- бути незначно гіршою за врахуванням інших критеріїв.

Графічно дане відношення зображають стрілкою від кращої до гіршої альтернативи або записом $A_i S A_k$, де A_i – переважаюча альтернатива.

Для визначення другої умови використовується попередньо визначений поріг та два набори порівнянь. Перший набір містить критерії альтернативи A_i, які є переважаючими відносно альтернативи A_k, а другий – критерії альтернативи A_i, які

поступаються критеріям альтернативи A_k. Дані набори називаються тестами на узгодженість та неузгодженість відповідно. Таким чином, методи групи ELECTRE окремо досліджують критерії, які надають та забирають перевагу альтернативи A_i над A_k. Дані тести представляють собою бінарні тести, які використовують індекси 0 та 1 для позначення проваленого та успішно пройденого тесту відповідно. Тест на неузгодженість має на меті оцінити наявність дуже високої протилежності між альтернативами та призначений для критеріїв, за якими A_i гірша порівняно з A_k.

За результатами виконання тестів, відношення двох альтернатив може перебувати у одному з трьох станів:

- A_i є переважаючою альтернативою над A_k ($A_i S A_k$): за умови якщо обидва тести були успішними.
- A_i є непорівнюваною з A_k ($A_i R A_k$): не існує відношення, у якому A_i або A_k є переважаючою над іншою альтернативою.
- A_i є нейтральною до A_k ($A_i I A_k$): одна альтернатива не має переваги над іншою.

2.3.2 Процес виконання вибору методом ELECTRE I

Оскільки серед методів сімейства ELECTRE саме ELECTRE I фокусується на вирішенні задачі вибору, то варто розглянути його детальніше. Для визначення переважаючих відношень між альтернативами у даному методі виконуються дев’ять послідовних кроків [31].

Першим кроком є нормалізація матриці вибору. Даний етап, як і у методі TOPSIS, необхідний для того, щоб трансформувати значення критеріїв альтернатив у порівнювану шкалу. Для виконання нормалізації використовується формула 2.5.

Другим етапом виконання методу ELECTRE I є врахування вагових коефіцієнтів у нормалізованій матриці вибору. Елементи матриці отримуються

шляхом обчислення добутку нормалізованого значення елемента та відповідного вагового коефіцієнта критерію. За результатами даного кроку буде отримано зважену матрицю вибору 2.6.

Третім кроком є визначення наборів узгодженості та неузгодженості. Даний крок має на меті розділити набір критеріїв на дві підмножини для кожної пари альтернатив. У загальному випадку набори узгодженості та неузгодженості можна подати формулами 2.12 та 2.13 відповідно.

$$C_{kl} = \{j | v_{kj} \geq v_{lj}, k \neq l\}, \tag{2.12}$$

$$D_{kl} = \{j | v_{kj} < v_{lj}, k \neq l\} = J - C_{kl} \tag{2.13}$$

Четвертим етапом виконання є побудова матриці узгодженості. На цьому кроці сума значень вагових коефіцієнтів, які пов'язані з відповідним набором узгодженості, розглядається як індекс узгодженості. Для кожного набору, використовуючи формулу 2.14, виконується обчислення даного індексу.

$$c_{kl} = \sum_{j \in C_{kl}} w_j \tag{2.14}$$

На основі обрахунку індексів узгодженості для всіх наборів формується матриця узгодженості *C*, яка у загальному випадку має наступний вигляд:

$$C = \begin{bmatrix} - & c_{12} & \cdots & c_{1m} \\ c_{21} & - & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & \cdots & c_{m(m-1)} & - \end{bmatrix} \tag{2.15}$$

На п'ятому кроці відбувається побудова матриці неузгодженості. Метою даного кроку є визначення ступеню, на який певна альтернатива гірше за іншу. Обчислення індексів неузгодженості виконується за допомогою формули 2.16.

$$d_{kl} = \frac{\max\limits_{j \in D_{kl}} |v_{kj} - v_{lj}|}{\max\limits_{j \in J} |v_{kj} - v_{lj}|} \tag{2.16}$$

Після обрахунку усіх індексів матриця невідповідності *D* будується подібним чином до матриці узгодженості:

$$D = \begin{bmatrix} - & d_{12} & \cdots & d_{1m} \\ d_{21} & - & \cdots & d_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & \cdots & d_{m(m-1)} & - \end{bmatrix} \quad (2.17)$$

У матриці 2.17 вищі значення d_{kl} показують, що A_k є менш доречним, ніж A_l. За нижчих значень навпаки – A_k є більш доречним від A_l. Значення у матриці неузгодженості варіюються від 0 до 1.

Шостим етапом методу ELECTRE I є визначення матриці індексу узгодженості. На цьому етапі враховується порогове значення індексу узгодженості, на основі якого визначається, чи може A_k переважати над A_l. Для цього значення c_{kl} порівнюється з певним пороговим значенням $\bar{c}$, і якщо $c_{kl} \geq \bar{c}$, то A_k є домінантною над A_l. Поріг може бути обчислений різними шляхами, наприклад, за допомогою формули 2.18, але зазвичай йому присвоюється значення рівне 0.7.

$$\bar{c} = \sum_{\substack{k=1 \\ k \neq l}}^{m} \sum_{\substack{l=1 \\ l \neq k}}^{m} \frac{c_{kl}}{m(m-1)} \quad (2.18)$$

Після визначення порогового значення будується булева матриця *F* за наступним рівнянням:

$$f_{kl} = \begin{cases} 0, c_{kl} < \bar{c} \\ 1, c_{kl} \geq \bar{c} \end{cases} \quad (2.19)$$

Кожен елемент зі значенням “1” у даній матриці означає, що дана альтернатива є переважаючою по відношенню до іншої альтернативи.

На сьомому кроці відбувається визначення матриці індексу неузгодженості. Подібний процес до побудови матриці індексу узгодженості використовується, щоб сформувати матрицю індексу неузгодженості *G*. Як і на попередньому етапі, порогове значення може бути обраховане за формулою 2.20, але зазвичай його встановлюють на рівні 0.3.

$$\bar{d} = \sum_{\substack{k=1 \\ k \neq l}}^{m} \sum_{\substack{l=1 \\ l \neq k}}^{m} \frac{d_{kl}}{m(m-1)} \quad (2.20)$$

Після визначення значення $\bar{d}$ також будується булева матриця використовуючи наступне рівняння:

$$g_{kl} = \begin{cases} 0, d_{kl} > \bar{d} \\ 1, d_{kl} \leq \bar{d} \end{cases} \quad (2.21)$$

Восьмим етапом застосування методу є побудова матриці загального домінування. Метою даного кроку є об'єднання матриць *F* та *G* шляхом обчислення їх перетину, який називають матрицею загального домінування *E*. Елементи даної матриці e_{kl} обчислюються за формулою 2.22.

$$e_{kl} = f_{kl} \times g_{kl} \quad (2.22)$$

Таким чином, елементи матриці загального домінування набувають булевих значень.

Останнім кроком методу ELECTRE I є виключення менш переважаючих альтернатив та формування ядра альтернатив. Ядро представляє собою набір альтернатив, які не поступаються жодній іншій альтернативі, але при цьому є переважаючими хоча б над однією іншою альтернативою. Для цього проводиться аналіз матриці *E*, отриманої на попередньому кроці. Якщо у даній матриці $e_{kl} = 1$, то це означає, що за матрицями узгодженості та неузгодженості альтернатива A_k є переважаючою над A_i. Однак, A_k може поступатися іншим альтернативам, тому для визначення ядра використовуються наступні умови:

$$\exists l \in \{1, \dots, m\}, k \neq l: e_{kl} = 1, \quad (2.23)$$

$$\forall i \in \{1, \dots, m\}, i \neq k: e_{ik} = 0 \quad (2.24)$$

Даний запис означає, що стовпець домінантної альтернативи повинен містити нулі, але при цьому в рядку даної альтернативи повинен знаходитися хоча б один елемент зі значенням 1. Таким чином, стовпці матриці *E*, які містять 1 є такими, які поступаються іншим альтернативам і не включаються до ядра. На рисунку 2.1 зображено приклад матриці *E* з ядром альтернатив (A_1, A_3).

$$E = \begin{array}{c} \begin{matrix} A_1 & A_2 & A_3 & A_4 \end{matrix} \\ \begin{bmatrix} - & 1 & 0 & 1 \\ 0 & - & 0 & 0 \\ 0 & 1 & - & 1 \\ 0 & 1 & 0 & - \end{bmatrix} \end{array} \begin{matrix} A_1 \\ A_2 \\ A_3 \\ A_4 \end{matrix}$$

Рисунок 2.1 – Приклад матриці *E* з ядром альтернатив [31]

Як згадувалося раніше у підрозділі 2.3.1, представлення матриці *E* ще можна подати графічно за допомогою стрілок, які починаються з переважаючої альтернативи та прямують в альтернативу, яка є гіршою за дану. Таке графічне представлення матриці *E* з рисунку 2.1 зображено на рисунку 2.2.

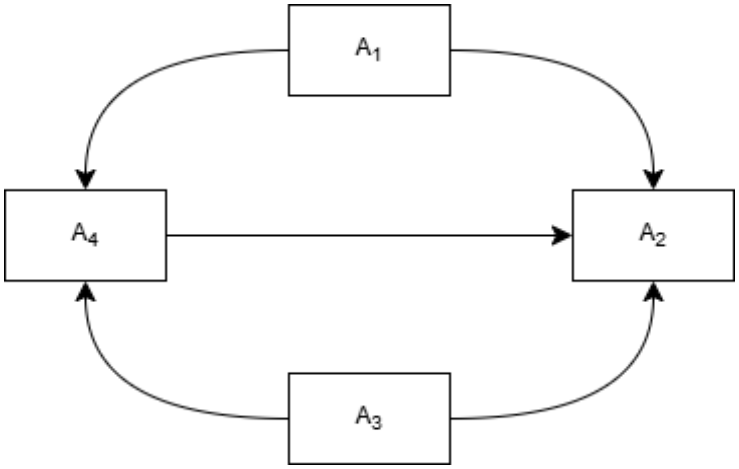

Рисунок 2.2 – Графічне представлення матриці *E* [31]

За результатами аналізу методів TOPSIS та сімейства методів ELECTRE, було обрано використовувати у роботі метод ELECTRE I. Даний підхід було обрано через його простоту та наочність, оскільки отримані результати можна легко зобразити графічно. Для застосування цього методу необхідна мінімальна кількість вхідних параметрів і водночас він дозволяє відокремити домінантні альтернативи, які не поступаються жодній іншій. На відміну від TOPSIS, який доцільніше використовувати для ранжування альтернатив, ELECTRE I фокусується на пошуку найсильніших кандидатів та формуванні ядра вибору.

2.4 Формування критеріїв вибору

У виборі інструменту для тестування безпеки API кожна організація орієнтується на власні пріоритети. Як стало зрозуміло з аналізу джерел у розділі 1.3.2, універсального набору критеріїв, які підійшли би всім без винятку, не існує. Залежно від масштабу, наявності ресурсів та технічної інфраструктури кожна компанія підбирає набір критеріїв, які є важливими саме їм. Тому для вибору інструменту тестування кібербезпеки Web API для системи електронної торгівлі було виділено наступні критерії вибору:

- Ціна.
- Інтеграція у CI/CD.
- Зручність застосування.
- Масштабованість.
- Типи тестування.
- Звітність та аналітика.

2.4.1 Ціна

Даний критерій визначає загальні фінансові витрати на придбання та експлуатацію інструменту тестування безпеки. Він включає не лише початкову вартість ліцензії чи підписки, а й усі супутні витрати, наприклад, навчання персоналу чи можливі додаткові модулі або розширення.

Багато комерційних рішень пропонують різні тарифні плани з додатковими функціями, тому важливим є порівняння інструментів за ціною та функціями доступних пакетів, щоб організації не довелося оплачувати функціонал інструменту, який не використовується. Також важливим моментом є масштабування компанії, оскільки часто за збільшення кількості залучених кінцевих точок доводиться доплачувати.

2.4.2 Інтеграція у CI/CD

Цей критерій означає здатність інструменту безпеки API вбудовуватися безпосередньо в процеси безперервної інтеграції та доставки, які використовуються командою розробки. Автоматичне виконання тестувань безпеки під час кожної спроби модифікувати програмний код веб-API дозволяє виявити на не допустити публікацію вразливостей.

Автоматизація тестування безпеки в CI/CD дозволяє надавати миттєвий зворотній зв’язок розробникам, що значно скорочує час на виправлення вразливостей та знижує ймовірність публікації небезпечного коду. Додатково вона забезпечує постійний моніторинг безпеки, що дозволяє відстежувати динаміку змін у стані безпеки, а також підтримувати історію результатів сканувань для аудиту.

2.4.3 Зручність застосування

Під цим критерієм розуміється легкість освоєння та використання інструменту, а саме: наскільки зрозумілим є інтерфейс, чи є готові шаблони для популярних сценаріїв, а також чи наявна детальна документація. Зручний інструмент дозволяє швидко налаштувати перші тести безпеки та мінімізувати часові витрати на навчання команди.

Зручний та інтуїтивно зрозумілий інтерфейс та легка структура налаштувань значно скорочують час впровадження, через що розробники можуть швидко розгорнути інструмент тестування API, запустити базові тести та отримати результат. Важливим елементом є наявність попередньо налаштованих шаблонів тестувань, що дає змогу почати роботу без глибокого вивчення особливостей засобу тестування безпеки.

2.4.4 Масштабованість

Цей критерій стосується здатності інструменту ефективно працювати при зростанні системи, наприклад, збільшенні кількості кінцевих точок чи трафіку. Масштабований засіб може запускати паралельні сканування або викликатися в контейнері чи кластері, щоб гарантувати швидкість перевірок без перевантаження ресурсів.

Системи електронної комерції зазвичай стрімко збільшуються, з’являються нові сервіси, функції тощо. Тому інструмент має підтримувати горизонтальне масштабування, щоб одночасно перевіряти десятки або сотні кінцевих точок API. Крім того, масштабованість передбачає можливість розширення функціоналу засобу, а саме: підключення нових модулів, плагінів або скриптів для запуску специфічних тестів без необхідності внесення модифікацій у програмний код інструменту. Це дає змогу адаптуватися до змін у архітектурі проєкту та швидко реагувати на нові загрози.

2.4.5 Типи тестування

Під цим критерієм мається на увазі різноманітність методів перевірки API на наявність вразливостей. Інструмент, який підтримує декілька типів сканування, наприклад, статичне та динамічне, дозволяє проводити комплексну оцінку безпеки прикладного програмного інтерфейсу.

Статичний аналіз дозволяє проводити аналіз коду та OpenAPI-контрактів, виявляючи помилки в описах кінцевих точок, невідповідності схем даних, відсутні поля, а також некеровані API. У той же час, динамічний аналіз його доповнює, виконуючи найбільш поширені атаки проти реальних прикладних програмних інтерфейсів. Особливо важливим елементом є можливість тестування автентифікації та авторизації, оскільки це дозволить імітувати запити з врахуванням ролей та рівнів доступу до ресурсів.

2.4.6 Звітність та аналітика

Цей критерій означає, наскільки детальні та зрозумілі звіти генерує інструмент, а також, які аналітичні можливості він надає. Наявність візуалізації трендів, статистики за знайденими вразливостями та фільтрація результатів допомагає оцінити поточний стан безпеки та планувати подальші заходи.

Після тестування інструмент повинен надавати докладну інформацію для розробників з зазначенням вразливих кінцевих точок, запитів, які викликали проблеми та рекомендації щодо їх виправлення. Важливим також є наявність інформаційної панелі, яка дозволяє відображати усю необхідну інформацію стосовно стану безпеки API.

Висновки до розділу 2

У даному розділі було розглянуто багатокритеріальне прийняття рішень та його методи. Було проаналізовано два популярні підходи вирішення задач вибору – TOPSIS та ELECTRE I, останній з яких було визначено більш доцільним до використання у вирішенні задачі вибору інструменту тестування кібербезпеки на основі аналізу Web API.

Також були сформовані критерії для вибору інструменту тестування безпеки, які будуть використані для оцінки відібраних інструментів, на основі чого буде обрано краще рішення шляхом використання методу ELECTRE I.

3 ЗАСТОСУВАННЯ МЕТОДУ ELECTRE I ДЛЯ ВИБОРУ ІНСТРУМЕНТУ ТЕСТУВАННЯ БЕЗПЕКИ ВЕБ-API

3.1 Формування вхідних даних для методу ELECTRE

Першим підготовчим кроком до застосування методу ELECTRE I є побудова матриці вибору, яка складається з альтернатив та критеріїв. У якості альтернатив було обрано розглянуті у першому розділі інструменти тестування безпеки на основі аналізу Web API: 42Crunch, Jit, Salt Security, Pynt, Postman & SoapUI. У другому розділі було обґрунтовано набір критеріїв вибору, а саме: ціна, інтеграція у CI/CD, зручність застосування, масштабованість, типи тестування, звітність та аналітика. Для зручності оцінки варто скласти порівняльну таблицю даних інструментів на основі їх огляду у підрозділі 1.3.1. Результати порівняння подано у таблиці 3.1.

Таблиця 3.1 – Порівняння інструментів тестування безпеки API

Інструмент / **Критерій**	**42Crunch**	**Jit**	**Salt Security**	**Pynt**	**Postman & SoapUI**
1	**2**	**3**	**4**	**5**	**6**
Ціна	Від $0 до $375+	Від $0 до $50+ за розробника, окрема оплата за DAST сканування	Від $3000 до $8333 за 5 та 100 млн. запитів на місяць	Від $0 до $11250+	Від $0 до $95+
Інтеграція у CI/CD	Готові плагіни для GitHub, GitLab, Jekins, Azure DevOps	CLI-агенти та плагіни для GitHub, GitLab, Jenkins, Azure DevOps	Часткові OpenAPI-аудити	Легка інтеграція за допомогою CLI	Newman та готові шаблони для GitHub, GitLab, Jenkins
Зручність застосування	Необхідні знання OpenAPI, зрозумілий інтерфейс користувача	Інтерфейс користувача з великою кількістю функцій	Складне налаштування віддзеркалення трафіку	CLI-орієнтований, швидке встановлення	Зрозумілі інтерфейси користувача

Кінець таблиці 3.1

1	2	3	4	5	6
Масшта-бованість	SaaS-архітектура, необмежена кількість сканувань	Необмежені статичні сканування	AI/ML-аналіз великої кількості запитів	Необхідність розгортання на декількох вузлах	Обмежені кількістю потоків, не-виконують паралельні сканування
Типи тестування	SAST, DAST	SAST, SCA, DAST, CSPM	Виявлення загроз під час виконання	DAST	Сканування за базовими правилами
Звітність та аналітика	Детальні звіти у PDF, HTML, JSON	Гнучкі інформаційні панелі, інтеграція з Jira	Інформаційна панель зі статистикою	CLI-звіти у JSON та CSV	Звіти у HTML, JSON, PDF з необхідністю фільтрування

На основі отриманої таблиці порівнянь потрібно провести їх оцінку за зазначеними критеріями. Оцінка інструментів проводилася за шкалою зі значеннями від 1 до 5 з урахуванням їх характеристик, для платформи електронної торгівлі середнього розміру, з незначною кількістю інженерів безпеки. Результати відображені у таблиці 3.2.

Таблиця 3.2 – Результати оцінювання інструментів

Інструмент / **Критерій**	**42Crunch**	**Jit**	**Salt Security**	**Pynt**	**Postman & SoapUI**
Ціна	3	2	1	2	4
Інтеграція у CI/CD	5	5	2	5	4
Зручність застосування	4	4	3	3	4
Масштабованість	4	4	5	3	2
Типи тестування	3	5	1	2	1
Звітність та аналітика	3	4	3	2	1

Другим кроком, який необхідно виконати перед початком застосування методу ELECTRE є визначення вагових значень критеріїв. Для цього було використано інтерв'ю головного інженера безпеки компанії «Meta» [32]. На основі висновків та тез з даного інтерв'ю кожному критерію було встановлено вагове значення у шкалі від 1 до 10. Результати визначення вагових значень критеріїв відображено у таблиці 3.3.

Таблиця 3.3 – Результати визначення вагових значень критеріїв

Критерій	**Вагове значення**
Ціна	3
Інтеграція у CI/CD	6
Зручність застосування	10
Масштабованість	5
Типи тестування	9
Звітність та аналітика	10

3.2 Виконання етапів оцінки альтернатив

Для проведення вибору інструменту тестування веб-API методом ELECTRE I було обрано використовувати бібліотеку pyDecision [33] написану на мові програмування Python. Дана бібліотека містить реалізацію великої кількості методів багатокритеріального прийняття рішень у вигляді функцій, серед яких наявне вищезгадане сімейство методів ELECTRE.

Як було описано у підрозділі 2.3.2, першим етапом є нормалізація матриці вибору та значень вагових критеріїв. Результати нормалізації було занесено у відповідні таблиці 3.4 та 3.5.

Таблиця 3.4 – Нормалізована матриця вибору

Інструмент / **Критерій**	**42Crunch**	**Jit**	**Salt Security**	**Pynt**	**Postman & SoapUI**
Ціна	0,25	0,17	0,08	0,17	0,33
Інтеграція у CI/CD	0,24	0,24	0,09	0,24	0,19
Зручність використання	0,22	0,22	0,17	0,17	0,22
Масштабованість	0,22	0,22	0,28	0,17	0,11
Типи тестування	0,25	0,42	0,08	0,17	0,08
Звітність та аналітика	0,23	0,31	0,23	0,15	0,08

Таблиця 3.5 – Вагові коефіцієнти критеріїв

Критерій	**Вагове значення**
Ціна	0,07
Інтеграція у CI/CD	0,14
Зручність використання	0,23
Масштабованість	0,12
Типи тестування	0,21
Звітність та аналітика	0,23

Наступним кроком є врахування вагових коефіцієнтів у матриці вибору та побудова на цій основі зваженої матриці вибору. Результати занесено у таблицю 3.6.

Таблиця 3.6 – Зважена матриця вибору

Інструмент / **Критерій**	**42Crunch**	**Jit**	**Salt Security**	**Pynt**	**Postman & SoapUI**
1	**2**	**3**	**4**	**5**	**6**
Ціна	0,0175	0,0119	0,0056	0,0119	0,0231
Інтеграція у CI/CD	0,0336	0,0336	0,0126	0,0336	0,0266

Кінець таблиці 3.6

1	2	3	4	5	6
Зручність використання	0,0506	0,0506	0,0391	0,0391	0,0506
Масштабованість	0,0264	0,0264	0,0336	0,0204	0,0132
Типи тестування	0,0525	0,0882	0,0168	0,0357	0,0168
Звітність та аналітика	0,0529	0,0713	0,0529	0,0345	0,0184

На 3-5 кроках відбувається визначення наборів узгодженості та неузгодженості і на їх основі формуються відповідні матриці. Результати даних кроків відображені у матрицях *C* та *D* відповідно:

$$C = \begin{bmatrix} 1 & 0{,}56 & 0{,}88 & 1 & 0{,}93 \\ 0{,}93 & 1 & 0{,}88 & 1 & 0{,}93 \\ 0{,}35 & 0{,}12 & 1 & 0{,}58 & 0{,}56 \\ 0{,}14 & 0{,}21 & 0{,}65 & 1 & 0{,}7 \\ 0{,}3 & 0{,}3 & 0{,}65 & 0{,}3 & 1 \end{bmatrix}, \quad (3.1)$$

$$D = \begin{bmatrix} 0 & 0{,}5 & 0{,}25 & 0 & 0{,}25 \\ 0{,}25 & 0 & 0{,}25 & 0 & 0{,}5 \\ 0{,}75 & 1 & 0 & 0{,}75 & 0{,}75 \\ 0{,}25 & 0{,}75 & 0{,}5 & 0 & 0{,}5 \\ 0{,}5 & 1 & 0{,}75 & 0{,}25 & 0 \end{bmatrix} \quad (3.2)$$

На шостому та сьомому кроках було використано стандартні індекси узгодженості та неузгодженості бібліотеки pyDecision, які встановлені на рівні 0,75 та 0,5 відповідно. На основі обчислених на попередніх кроках матриць 3.1 та 3.2 було сформовано булеві матриці *F* та *G*:

$$F = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (3.3)$$

$$G = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (3.4)$$

3.3 Результати проведеної оцінки

Відповідно до алгоритму ELECTRE I, останнім кроком проведення оцінки є об'єднання матриць 3.3 та 3.4. Таким чином, за результатами виконання коду було отримано матрицю загального домінування *E*.

$$E = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.5)$$

З аналізу матриці можна зробити висновок, що альтернатива A_2 (Jit) формує ядро вибору, оскільки у відповідному стовпці наявні лише нулі, а у другому рядку є мінімум одна одиниця. Тому лише альтернатива A_2 виконує умови для входження у ядро вибору. Для зручності матрицю *E* було подано у вигляді графа, зображеного на рисунку 3.1, а альтернативи позначені наступним чином:

- A_1 – 42Crunch.
- A_2 – Jit.
- A_3 – Salt Security.
- A_4 – Pynt.
- A_5 – Postman & SoapUI.

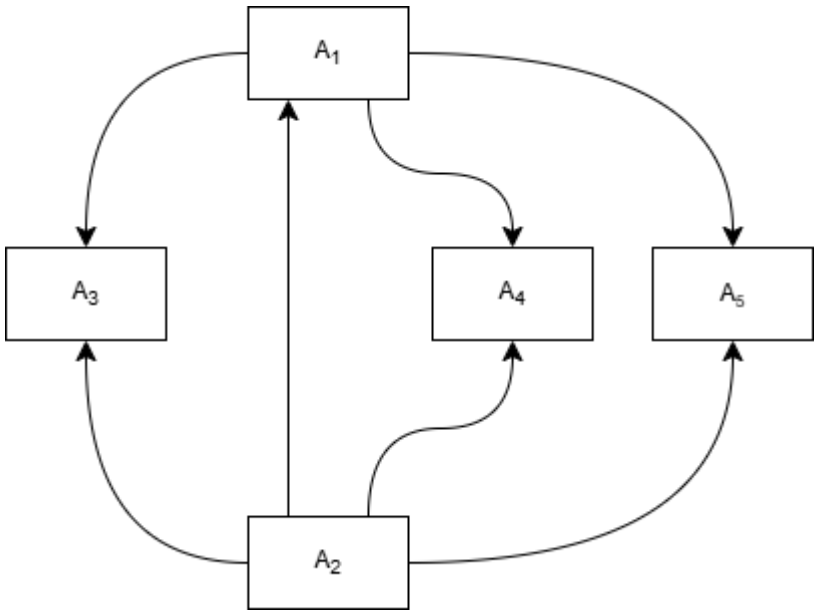

Рисунок 3.1 – Граф результатів проведеної оцінки методом ELECTRE I

На основі матриці загального домінування та графу результатів можна встановити наступні відношення альтернатив: $A_1 \to A_3, A_1 \to A_4, A_1 \to A_5, A_2 \to A_1,$ $A_2 \to A_3, A_2 \to A_4, A_2 \to A_5$. Таким чином, альтернативи A_3, A_4, A_5 не є ядерними альтернативами, оскільки вони поступаються A_1 та A_2, але при цьому A_2 переважає альтернативу A_1.

Тому, за результатами виконання методу ELECTRE I, найкращим вибором є інструмент тестування безпеки API Jit. Він не поступається жодному іншому засобу за обраними критеріями та формує ядро вибору.

Висновки до розділу 3

У даному розділі було виконано порівняння розглянутих інструментів тестування безпеки на основі аналізу Web API. Результати, з врахуванням експертної думки та особливостей інструментів, були подані у вигляді двох таблиць, які містять короткий опис та оцінку таких альтернатив, як 42Crunch, Jit, Salt Security, Pynt та Postman & SoapUI за встановленими критеріями. Інструменти порівнювалися за шістьма критеріями: ціна, інтеграція у CI/CD, зручність використання, масштабованість, типи тестування, звітність та аналітика.

На основі сформованих матриці вибору та вагових коефіцієнтів було описано кроки виконання багатокритеріального прийняття рішень методом ELECTRE I. У результаті було отримано матрицю загального домінування за якою було побудовано граф результатів проведеної оцінки. Аналіз матриці та графу показав, що інструмент тестування Jit є найоптимальнішим вибором серед розглянутих засобів тестування кібербезпеки API.

ВИСНОВКИ

У ході дипломної роботи було проведено огляд сучасного стану безпеки Web API у сфері електронної торгівлі. За результатами аналізу стало зрозумілим, що проблема безпеки інтерфейсів є досить поширеною, оскільки значна частина веб-атак проти комерційних організацій була спрямована саме на прикладні програмні інтерфейси. Також було визначено основні загрози для API, зокрема: ін’єкційні атаки, неправильна автентифікація та управління сесіями, небезпечна передача даних, атаки відмови в обслуговуванні, неправомірне використання ключів API, недостатня перевірка вхідних даних, неконтрольовані перенаправлення та переспрямування, неконтрольовані перенаправлення вхідних даних, відсутність контролю доступу, відсутність моніторингу та логування.

Було проаналізовано доступні на ринку інструменти тестування безпеки API: 42Crunch, Jit, Salt Security, Pynt, Postman & SoapUI, а також характерні особливості кожного з них. Результати аналізу показали, що наявні засоби тестування можуть досить сильно відрізнятися один від одного, тому перед організаціями у сфері електронної торгівлі постає проблема вибору оптимального інструменту.

Для вирішення даної проблеми було розглянуто підхід багатокритеріального прийняття рішень. На основі огляду існуючих MCDM-методів були детальніше розглянуті такі представники, як TOPSIS та сімейство методів ELECTRE. Після аналізу їхніх переваг, недоліків та алгоритмів виконання, було обрано використовувати метод ELECTRE I, оскільки він забезпечує зручну та зрозумілу візуалізацію результатів, простий алгоритм виконання, а також не є компенсаційним методом.

На основі аналізу джерел було сформовано набір критеріїв, які є важливими для системи електронної торгівлі, а саме: ціна, інтеграція у CI/CD, зручність використання, масштабованість, типи тестування, звітність та аналітика.

Після порівняння засобів тестування API за визначеними критеріями, було сформовано матрицю вибору та вагові коефіцієнти критеріїв. Вони були застосовані як вхідні дані для методу ELECTRE I, що дозволило отримати матрицю загального домінування та граф результатів проведеної оцінки. З аналізу відношень у матриці та графі було отримано, що ядро вибору формує лише альтернатива A_2 (Jit), що робить її найоптимальнішим вибором серед розглянутих засобів. Таким чином, було практично продемонстровано використання даного підходу для вибору інструменту тестування безпеки на основі аналізу Web API, що дозволяє розробникам, інженерам безпеки або адміністраторам систем електронної торгівлі здійснити вибір засобу, виходячи з потреб та обмежень їх організації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Check Point Team. A Closer Look at Q3 2024: 75% Surge in Cyber Attacks Worldwide. [Електронний ресурс] // Check Point. – Режим доступу до ресурсу: https://blog.checkpoint.com/research/a-closer-look-at-q3-2024-75-surge-in-cyber-attacks-worldwide/.

2. Гальчинський Л. Ю., Носок С. О., Васильченко А. А. Імітаційна модель оцінки механізму державних закупівель з метою підвищення їх ефективності [Текст] // Адаптивні системи автоматичного управління. – 2010. – Т. 16. – С. 23-32. – Режим доступу до ресурсу: http://nbuv.gov.ua/UJRN/asau_2010_16_7.

3. Why is Robust API Security Crucial in eCommerce?. [Електронний ресурс] // The Hacker News. – Режим доступу до ресурсу: https://thehackernews.com/2022/12/why-is-robust-api-security-crucial-in.html.

4. 2024 API Security Impact Study. Retail and Ecommerce Industry. [Електронний ресурс] // Akamai. – Режим доступу до ресурсу: https://www.akamai.com/site/en/documents/brochure/2024/akamai-2024-api-security-impact-study-retail-and-ecommerce-industry.pdf.

5. Web API basics – definition, technologies, features, and uses. [Електронний ресурс] // Gravitee. – Режим доступу до ресурсу: https://www.gravitee.io/understanding-web-apis-and-their-use.

6. Demir D. What Are the Components of an API?. [Електронний ресурс] //Apidog. – Режим доступу до ресурсу: https://apidog.com/blog/what-are-the-components-of-an-api/.

7. What are the components of an API?. [Електронний ресурс] // Postman. – Режим доступу до ресурсу: https://blog.postman.com/what-are-the-components-of-an-api/.

8. What Is an API Gateway?. [Електронний ресурс] // F5. – Режим доступу до ресурсу: https://www.f5.com/glossary/api-gateway.

9. A guide to the different types of APIs. [Електронний ресурс] // Postman. – Режим доступу до ресурсу: https://blog.postman.com/different-types-of-apis/.

10. Shepard A. What are The Different Types of APIs and Protocols?. [Електронний ресурс] // Kong Inc. – Режим доступу до ресурсу: https://konghq.com/blog/learning-center/different-api-types-and-use-cases.

11. What Is API Security?. [Електронний ресурс] // Akamai. – Режим доступу до ресурсу: https://www.akamai.com/glossary/what-is-api-security.

12. Engel G. API Security: The New North, South, East and West of Cybersecurity. [Електронний ресурс] // Cyber Protection Magazine. – Режим доступу до ресурсу: https://cyberprotection-magazine.com/api-security-the-new-north-south-east-and-west-of-cybersecurity.

13. What Are API Security Risks?. [Електронний ресурс] // Akamai. – Режим доступу до ресурсу: https://www.akamai.com/glossary/what-are-api-security-risks.

14. Mendo T. Balancing Efficiency and Security: API Protection in E-commerce. [Електронний ресурс] // Snyk API & Web. – Режим доступу до ресурсу: https://probely.com/blog/balancing-efficiency-and-security-api-protection-in-e-commerce.

15. API Security Testing: Essential Tools & Checklist for eCommerce Success. [Електронний ресурс] // Blogs @ Mindfire Solutions. – Режим доступу до ресурсу: https://www.mindfiresolutions.com/blog/2025/04/api-security-testing-for-ecommerce-success/.

16. Forêt C. ISO 27005: Everything you need to know if you are considering implementing it. [Електронний ресурс] // Cyber Risk Quantification & Cybersecurity Management Solutions | C-Risk. – Режим доступу до ресурсу: https://www.c-risk.com/blog/iso-27005.

17. Ю. І. Пташник, Л. Ю. Гальчинський. Оцінка кіберризиків API у сфері електронної торгівлі. [Текст] // Теоретичні і прикладні проблеми фізики, математики та інформатики – 2025 – С. 252-254 – Режим доступу до ресурсу: http://conf.ipt.kpi.ua/arxiv/2025-2/.

18. Tal L. API Security Testing: 6 Things To Keep In Mind. [Електронний ресурс] // Snyk. – Режим доступу до ресурсу: https://snyk.io/articles/application-security/api-security/api-security-testing/.

19. 42Crunch API Security Platform. [Електронний ресурс] // 42Crunch. – Режим доступу до ресурсу: https://docs.42crunch.com/latest/content/concepts/about_platform.htm.

20. What OWASP ZAP can do, and when to use it. Jit. [Електронний ресурс] – Режим доступу до ресурсу: https://www.jit.io/resources/owasp-zap.

21. API Security Platform - Complete API Protection Platform. API Security Solutions & Tools - API Protection Platform. [Електронний ресурс] – Режим доступу до ресурсу: https://salt.security/platform.

22. Security Testing Overview | Documentation. Why API Security is Critical? | Documentation. [Електронний ресурс] – Режим доступу до ресурсу: https://docs.pynt.io/documentation/api-security-testing/security-testing-overview.

23. Postman: The World's Leading API Platform. [Електронний ресурс] // Postman API Platform. – Режим доступу до ресурсу: https://www.postman.com/.

24. Getting Started with Security Testing. [Електронний ресурс] // SoapUI. – Режим доступу до ресурсу: https://www.soapui.org/docs/security-testing/getting-started/.

25. Kuber P. What are API Security Scanners and How to Choose the Right One?. [Електронний ресурс] // Astra. – Режим доступу до ресурсу: https://www.getastra.com/blog/api-security/what-are-api-security-scanners/.

26. The Ultimate Guide to Choosing an API Testing Framework. [Електронний ресурс] // StackHawk. – Режим доступу до ресурсу: https://www.stackhawk.com/blog/api-testing-framework/.

27. Taherdoost H., Madanchian M. Multi-Criteria Decision Making (MCDM) Methods and Concepts. [Текст] // Encyclopedia. 2023. Т. 3, № 1. С. 77–87. – Режим доступу до ресурсу: https://doi.org/10.3390/encyclopedia3010006.

28. Madanchian M., Taherdoost H. A comprehensive guide to the TOPSIS method for multi-criteria decision making. [Текст] // Sustainable Social Development. 2023. Т. 1, № 1. – Режим доступу до ресурсу: https://doi.org/10.54517/ssd.v1i1.2220.

29. Galchynsky, L., Graivoronskyi, M., Dmytrenko, O.: evaluation of machine learning methods to detect DoS / DDoS attacks on IoT. [Текст] // In: CEUR Workshop Proceedings, 3241, 225–236 (2021) – Режим доступу до ресурсу: https://ceur-ws.org/Vol-3241/paper21.pdf.

30. An Overview of ELECTRE Methods and their Recent Extensions / J. R. Figueira та ін. [Текст] // Journal of Multi-Criteria Decision Analysis. 2012. Т. 20, № 1-2. С. 61–85. – Режим доступу до ресурсу: https://doi.org/10.1002/mcda.1482.

31. Taherdoost H., Madanchian M. A Comprehensive Overview of the ELECTRE Method in Multi Criteria Decision-Making. [Текст] // Journal of Management Science & Engineering Research. 2023. Т. 6, № 2. – Режим доступу до ресурсу: https://doi.org/10.30564/jmser.v6i2.5637.

32. Charikova A. Lack of effective DAST tools Aleksandr Krasnov (Meta). [Електронний ресурс] // Escape DAST - Application Security Blog. – Режим доступу до ресурсу: https://escape.tech/blog/lack-of-effective-dast-the-elephant-in-appsec/.

33. PEREIRA, V.; BASILIO, M.P.; SANTOS, C.H.T (2024). Enhancing Decision Analysis with a Large Language Model: pyDecision a Comprehensive Library of MCDA Methods in Python. [Текст] // arXiv. – Режим доступу до ресурсу: https://arxiv.org/abs/2404.06370.

ДОДАТОК А PYTHON 3 СКРИПТ TOOL_SELECTION.PY

```python
from numpy import array
from pyDecision.algorithm import electre_i

decision_matrix = array([
    # C1 C2 C3 C4 C5 C6
    [3, 5, 4, 4, 3, 3],  # A1
    [2, 5, 4, 4, 5, 4],  # A2
    [1, 2, 3, 5, 1, 3],  # A3
    [2, 5, 3, 3, 2, 2],  # A4
    [4, 4, 4, 2, 1, 1]  # A5
])

W = [0.07, 0.14, 0.23, 0.12, 0.21, 0.23]

concordance, discordance, dominance, kernel, dominated = electre_i(dataset=decision_matrix,
                                                                   W=W,
                                                                   remove_cycles=True,
                                                                   graph=False)

print(concordance)
print()
print(discordance)
print()
print(dominance)
print()
print(kernel)
```