МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

15.03 — КМР. 1939–“С” 2022.12.30. 27 ПЗ

ОСАДЧОГО ДЕНИСА ТАРАСОВИЧА

2023 р.

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ
І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет інформаційних технологій

УДК 004.02

«ПОГОДЖЕНО»	**«ДОПУСКАЄТЬСЯ ДО ЗАХИСТУ»**
Декан факультету інформаційних технологій	Завідувач кафедри комп'ютерних наук
Глазунова О.Г., д.п.н., професор	Голуб Б.Л., к.т.н., доцент
_____	_____
____ _____ 2023 р.	**____ _____ 2023 р.**

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему Технологія виявлення мережевих атак малої інтенсивності

Спеціальність Інженерія програмного забезпечення
(код і назва)

Освітня програма Програмне забезпечення інформаційних систем
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Гарант освітньої програми

кандидат ф.-м. наук, доцент _____ Кириченко В.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Керівник магістерської кваліфікаційної роботи

доктор т.н., професор _____ Семко В.В.
(науковий ступінь та вчене звання) (підпис) (ПІБ)

Виконав _____ Осадчий Денис Тарасович
(підпис) (ПІБ студента)

КИЇВ-2023

НАЦІОНАЛЬНИЙ УНІВЕРСТИТЕТ БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ УКРАЇНИ

Факультет (ННІ) Інформаційних технологій

ЗАТВЕРДЖУЮ
Завідувач кафедри_____
к.т.н., доцент _____ Голуб Б.Л.
(науковий ступінь, вчене звання) (підпис) (ПІБ)
"____" _____ 20____ року

З А В Д А Н Н Я

ДО ВИКОНАННЯ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ СТУДЕНТУ

Осадчий Денис Тарасович
(прізвище, ім'я, по батькові)

Спеціальність 121 Інженерія програмного забезпечення
(код і назва)

Освітня програма Інформаційні управляючі системи та технології
(назва)

Орієнтація освітньої програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Тема магістерської кваліфікаційної роботи Технологія виявлення мережевих атак малої інтенсивності

затверджена наказом ректора НУБіП України від "30" грудня 2022 р. № 1939-"С"

Термін подання завершеної роботи на кафедру "5" листопада 2023
(рік, місяць, число)

Вихідні дані до магістерської кваліфікаційної роботи _____

Перелік питань, що підлягають дослідженню:

1. Аналіз та оцінка впливу сучасних технологій на ефективність систем виявлення мережевих атак
2. Визначити основні принципи проектування та структури сучасних систем виявлення мережевих атак
3. Розробка ефективного методу для аналізу та виявлення мережевих атак малої інтенсивності
4. Розробка архітектури та інтеграція технології виявлення в сучасні мережеві середовища
5. Експериментальне випробування розробленої технології та оцінка її ефективності

Перелік графічного матеріалу (за потреби) _____

Дата видачі завдання "30" грудня 2022 р.

Керівник магістерської кваліфікаційної роботи _____ Семко В.В.
(підпис) (прізвище та ініціали)

Завдання прийняв до виконання _____ Осадчий Д.Т
(підпис) (прізвище та ініціали студента)

ЗМІСТ

ВСТУП 5

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 7

1.1 Огляд сучасного стану кібербезпеки 7

1.2. Сучасні типи технологій виявлення мережевих атак 8

1.3. Аналіз можливостей IDPS систем та їх методик виявлення 12

1.4. Порівняння готових рішень систем виявлення вторгненнь 16

1.5. Формулювання завдань дослідження 17

2. МОДЕЛЮВАННЯ СИСТЕМИ 19

2.1. Поверхневе моделювання системи 19

2.2 Дослідження способів аналізу атак за допомогою штучного інтелекту 27

3. РОЗРОБКА АЛГОРИТМІВ IDPS ВЕБ-СИСТЕМИ 30

3.1. Вибір інструментів для розробки 30

3.2. Архітектура бібліотеки 31

3.3. Принцип роботи трекера дій 35

3.5. Вибір інструментів для розробки 47

3.6. Практична реалізація розробленого методу 48

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ 54

4.1. Потенційна сфера застосування 54

4.2. Алгоритм створення гітарного тренажера..... 55

ВИСНОВКИ 57

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ 59

ВСТУП

В сучасному цифровому світі, де інформаційні технології стають невід'ємною частиною багатьох аспектів бізнесу та повсякденного життя, проблема кібербезпеки набуває все більшої актуальності. Однією з важливих складових кібербезпеки є технології виявлення мережевих атак, особливо атак малої інтенсивності, які можуть залишатися непоміченими традиційними системами виявлення.

Стрімкий ріст технологій та еволюція методів атак вже багато років викликають ефект «перегонів озброєнь» і вимагають своєчасного, а саме вдалого вдосконалення технологій виявлення для забезпечення ефективного та надійного захисту мережевого простору. Слідуючи з цього, атаки різного роду всеодно, рано чи пізно, зможуть проникнути навіть через найсучасніші на момент заходи безпеки і виникає гостра потреба в розробці нових методів та технологій для їх вчасного виявлення та запобігання.

Також, ретельну увагу потрібно приділити саме сферам використання та цільовим аудиторіям продуктів систем виявлення мережевих атак, адже лише дуже низький відсоток користувачів мають ресурси для використання дійсно просунутих та високо налаштовуваних рішень захисту, а саме великий бізнес та державні установи. Потрібно ураховувати потреби у захисті звичайних користувачів, адже саме вони найчастіше і стають цілями шкідливого програмного забезпечення, що вільно пересувається по мережі.

Метою даного дослідження є ретельний аналіз та розробка технології виявлення мережевих атак малої інтенсивності, спрямованої на забезпечення надійного функціонування в умовах постійно зростаючого ризику кібератак. Для досягнення цієї мети передбачено вирішення наступних завдань:

- Аналіз та оцінка впливу сучасних технологій на ефективність систем виявлення мережевих атак.
- Визначити основні принципи проектування та структури сучасних систем виявлення мережевих атак.
- Розробка ефективного методу для аналізу та виявлення мережевих атак малої інтенсивності.
- Розробка архітектури та інтеграція технології виявлення в сучасні мережеві середовища.
- Експериментальне випробування розробленої технології та оцінка її ефективності.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд сучасного стану кібербезпеки

У сучасному світі, де веб-орієнтовані технології стають необхідною складовою нашого повсякденного існування, виникає проблема забезпечення ефективної кібербезпеки. Збільшення кількості та складності кіберзагроз вимагає комплексних підходів до захисту від кіберзлочинів. Необхідно враховувати тенденції у кіберзлочинах, які не обмежуються лише збільшенням кількості атак. Розвиток нових методів та тактик кіберзлочинців ставить під сумнів традиційні методи захисту, вимагаючи вдосконалення систем виявлення вторгнень та стратегій реагування.

Національні та міжнародні організації активно працюють над визначенням стратегій кібербезпеки, спрямованих на забезпечення стійкості інфраструктури та конфіденційності інформації. Ефективні заходи з кібербезпеки включають в себе використання інтелектуальних систем виявлення вторгнень, а також розвиток алгоритмів захисту. Разом із тим, компанії та організації впроваджують передові технології, такі як штучний інтелект, блокчейн та квантові обчислення, для забезпечення більш ефективного контролю над кіберзагрозами. Однак це також викликає необхідність постійного оновлення та адаптації до нових викликів.

Сучасний стан кібербезпеки відображає суттєві тенденції, виклики та перспективи у галузі, визначаючи потребу в інноваційних та ефективних підходах до захисту від кіберзагроз, адже навіть кіберзаходи, проведені державами, стають невід'ємною частиною геополітичних конфліктів. Зростання участі держав у кіберпросторі ставить перед кібербезпекою нові завдання та вимагає вдосконалених стратегій захисту.

1.2. Сучасні типи технологій виявлення мережевих атак

Номінально, сучасними представленнями сутностей, що мають можливість виявити та запобігти мережевій атаці у кібербезпеці є IDS (Intrusion Detection System) та IPS (Intrusion Prevention System). IDS створене саме для пошуку та виявлення загрози аналітичними шляхами, а IPS виконує роль блокатора загрози. Найчастішим варіантом використання обох технологій є їх об'єднання у вигляді IPDS (Intrusion Detection and Prevention System). В цьому випадку, система стає майже досконалим механізмом захисту, що обмежується лише ресурсами та функціоналом виділеним на пошук загроз. Тому, розглянемо саме системи пошуку загроз у мережі.

Система виявлення вторгнень (IDS) – це, як правило, програмне забезпечення або апаратний пристрій, який відстежує вхідний і вихідний мережевий трафік на наявність ознак зловмисної діяльності або порушень політики безпеки. Системи виявлення вторгнень і продукти IDS часто порівнюють із сигналізацією зловмисника, яка сповіщає вас про будь-які дії, які можуть поставити під загрозу ваші дані або мережу.

Продукти IDS шукають підозрілу поведінку або ознаки потенційної компрометації, аналізуючи пакети, які переміщуються у вашій мережі, і моделі мережевого трафіку, щоб виявити будь-які аномалії. Системи виявлення вторгнень зазвичай пасивні за своєю природою, хоча деякі системи виявлення вторгнень (IPDS) можуть діяти, коли виявляють зловмисну поведінку. Однак загалом вони в основному використовуються для досягнення видимості в режимі реального часу випадків потенційного зламу мережі.

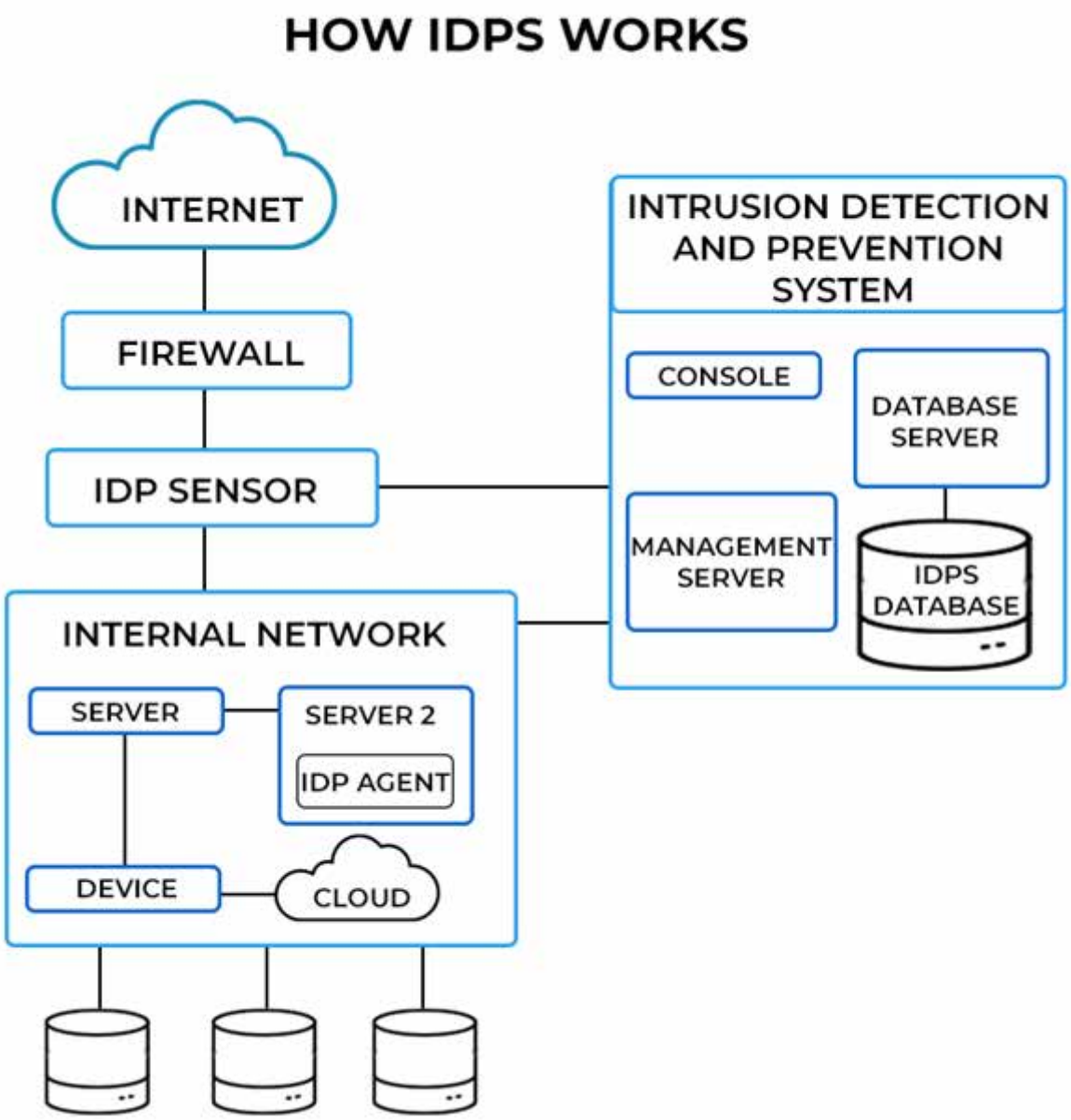

Рисунок 1.1 Схематичне представлення роботи IDPS системи

Залежно від типу системи виявлення вторгнень, яка була розгорнута, різні продукти IDS поводитимуться по-різному. Наприклад, мережева система виявлення вторгнень (NIDS) стратегічно розмістить датчики в кількох місцях у самій мережі. Потім ці датчики відстежуватимуть мережевий трафік, не створюючи проблем з продуктивністю або вузьких місць. З іншого боку, системи виявлення вторгнень на основі хостів (HIDS) працюють на певних пристроях і

хостах і здатні контролювати трафік лише для цих конкретних пристроїв і хостів. У цілому, організації можуть розглянути впровадження чотирьох типів систем виявлення та запобігання вторгненням залежно від типу розгортання, який вони шукають.

- **Мережева система запобігання вторгненням** (NIPS): мережеві системи запобігання вторгненням відстежують цілі мережі або сегменти мережі на наявність зловмисного трафіку. Зазвичай це робиться шляхом аналізу активності протоколу. Якщо активність протоколу збігається з базою даних відомих атак, відповідна інформація не може пройти. NIPS зазвичай розгортають на межах мережі, за брандмауерами, маршрутизаторами та серверами віддаленого доступу.
- **Система запобігання бездротовому вторгненню** (WIPS). Системи запобігання бездротовому вторгненню відстежують бездротові мережі, аналізуючи протоколи окремих бездротових мереж. Хоча WIPS є цінним у діапазоні бездротової мережі організації, ці системи не аналізують вищі мережеві протоколи, такі як протокол керування передачею (TCP). Системи запобігання бездротовому вторгненню розгортаються всередині бездротової мережі та в зонах, чутливих до неавторизованого бездротового підключення.
- **Система аналізу поведінки мережі** (NBA): у той час як NIPS аналізує відхилення в активності протоколу, системи аналізу поведінки мережі визначають загрози, перевіряючи незвичні шаблони трафіку. Такі шаблони, як правило, є результатом порушень політики, атак, створених зловмисним програмним забезпеченням , або атак розподіленої відмови в обслуговуванні (DDoS). Системи NBA розгортаються у внутрішніх мережах організації та в точках, де трафік проходить між внутрішньою та зовнішньою мережами.

- **Система запобігання вторгненням на основі хоста** (HIPS): системи запобігання вторгненню на основі хоста відрізняються від інших тим, що вони розгорнуті на одному хості. Ці хости є критично важливими серверами з важливими даними або загальнодоступними серверами, які можуть стати шлюзами до внутрішніх систем. HIPS відстежує трафік, що надходить і виходить із цього конкретного хоста, відстежуючи запущені процеси, мережеву активність, системні журнали, діяльність програми та зміни конфігурації.

Тип IDPS	**Тип розгортання**	**Виявлені типи активності**
NIPS	Межі мережі, за брандмауером і маршрутизаторами та серверами віддаленого доступу	Діяльність мережевого, транспортного та додаткового рівня TCP/IP
WIPS	У межах бездротової мережі	Активність бездротового протоколу, неавторизоване використання WLAN
NBA	Внутрішні мережі та в точках, де трафік проходить між внутрішньою та зовнішньою мережами	Активність мережевого, транспортного та програмного рівня TCP/IP з аномаліями на рівні протоколу

Тип IDPS	Тип розгортання	Виявлені типи активності
HIPS	Індивідуальні хости: критичні сервери або загальнодоступні сервери	Діяльність хост-програми та операційної системи (ОС); мережевий, транспортний та додатковий рівень TCP/IP

1.3. Аналіз можливостей IDPS систем та їх методик виявлення

Системи IDP мають два рівні широких функціональних можливостей — виявлення та запобігання. На кожному рівні більшість рішень пропонують деякі основні підходи:

1. **Пороговий моніторинг.** Перший крок моніторингу порогових значень полягає у встановленні прийнятних рівнів, пов’язаних з кожним користувачем, програмою та поведінкою системи. Приклади показників, які використовуються під час моніторингу порогового значення, включають кількість невдалих спроб входу, кількість завантажень із певного джерела або навіть щось трохи складніше, наприклад прийнятний час доступу до певного ресурсу. Система моніторингу сповіщає адміністраторів і іноді запускає автоматичні відповіді, коли перевищено порогове значення.

Лише наявність порогового моніторингу замість виявлення вторгнень має власний набір проблем. Найчастіше складну інфраструктуру, що лежить в основі

операцій і пропозицій організації, неможливо відфільтрувати за кількома показниками. Ці порогові значення також мають тенденцію змінюватися в міру зростання клієнтської бази та послуг компанії. Дуже суворе впровадження порогового моніторингу в цих випадках може спричинити багато помилкових спрацьовувань. Помилковий позитивний результат у контексті рішень IDP – це коли доброякісна активність визначається як підозріла.

2. **Профілювання.** Системи виявлення та запобігання вторгненням пропонують два типи профілювання: профілювання користувача та профілювання ресурсів.

Профілювання користувачів передбачає моніторинг того, чи користувач із певною роллю чи групою користувачів генерує лише дозволений трафік. Наприклад, лише користувач DevOps може мати доступ до додатків на хмарному сервері. Програміст може отримати доступ до даних лише в серверному середовищі ізольованого програмного середовища. Короткостроковий моніторинг профілів користувачів дозволяє адміністраторам переглядати останні моделі роботи, тоді як довгострокове профілювання забезпечує розширене уявлення про використання ресурсів. Це стане в нагоді під час створення бази для нормальної поведінки та для створення самої ролі користувача.

Профілювання ресурсів визначає, як кожна система, хост і програма споживає та генерує дані. Програма з раптово збільшеним робочим процесом може вказувати на зловмисну поведінку.

Профілювання виконуваного файлу повідомляє адміністраторам, які програми зазвичай встановлюються та запускаються окремими користувачами, програмами та системами. Наприклад, на хості може бути запущена програма, яка має доступ лише до певних файлів. Будь-який інший файл або запит до шахрайської бази даних свідчить про нечесну гру. Такий тип профілювання

дозволяє легко відстежувати помилково завантажене шкідливе програмне забезпечення, програми-вимагачі або трояни.

Іноді профілювання може ускладнити інтерпретацію загального мережевого трафіку та нерівностей, які з ним виникають. Найкраще для профілювання лежить між надто широкими профілями, які дозволяють поганим акторам, і надто вузькими, які заважають продуктивності.

Також, сучасні системи виявлення мережевих атак поділяються на три основні категорії методик виявлення, а саме:

1. **Виявлення на основі сигнатур.** Сигнатура — це певний шаблон у корисному навантаженні. Цей конкретний шаблон може бути будь-яким: від послідовності 1 і 0 до кількості байтів. Більшість зловмисних програм і кібератак мають власний «підпис», який можна ідентифікувати. Іншим прикладом підпису є щось таке просте, як назва вкладення у шкідливому електронному листі.

Система IDP підтримує базу даних сигнатур відомих зловмисних програм із виявленням на основі сигнатур. Щоразу, коли з'являється нове шкідливе програмне забезпечення, ця база даних оновлюється. Система виявлення працює, перевіряючи корисне навантаження трафіку за цією базою даних і сповіщаючи про збіг.

Виявлення на основі сигнатур очевидно не може працювати, якщо зловмисне програмне забезпечення невідоме раніше. Він не перевіряє природу корисного навантаження та не може надати адміністраторам таку інформацію, як попередній запит на зловмисну відповідь.

2. **Виявлення аномалій.** Виявлення аномалій працює над пороговим моніторингом і профілюванням. Налаштовано «нормальну» поведінку всіх користувачів, хостів, систем і програм. Будь-яке відхилення від цієї норми вважається аномалією і про нього попереджають. Наприклад, якщо ідентифікатор електронної пошти генерує сотні електронних листів протягом

кількох годин, ймовірність того, що обліковий запис електронної пошти буде зламано, висока.

Виявлення аномалій краще, ніж виявлення на основі сигнатур, якщо розглядати нові атаки, яких немає в базі даних сигнатур. Створення цих базових профілів займає багато часу (також відомого як «період навчання»). Навіть тоді показники помилкових спрацьовувань можуть бути високими, особливо в динамічному середовищі.

3. **Аналіз протоколу з підтримкою станів.** Виявлення аномалій використовує профілі хоста або мережі для визначення підозрілої активності. Аналіз протоколу з підтримкою стану йде ще далі й використовує попередньо визначені стандарти кожного стану протоколу для перевірки відхилень.

Наприклад, протокол передачі файлів (FTP) дозволяє входити лише в разі неавтентифікації. Після автентифікації сеансу користувачі можуть переглядати, створювати або змінювати файли на основі своїх дозволів. Ця інформація є частиною визначення протоколу FTP. Система виявлення вторгнень аналізує відповідність цим нормам. Такий аналіз протоколу з відстеженням стану дозволяє легко відстежувати автентифікатор у кожному сеансі та подальшу діяльність, пов’язану з цим запитом.

Аналіз протоколу з підтримкою стану значною мірою залежить від визначень протоколів, керованих постачальником. Деталізація означає, що він також вимагає ресурсів, займаючи дорогоцінну пропускну здатність під час відстеження одночасних сеансів. Кожна з цих технік або забезпечує запобігання вхідним атакам, або допомагає адміністраторам виявляти вразливі місця системи безпеки. Більшість рішень IDP пропонують поєднання кількох підходів.

1.4. Порівняння готових рішень систем виявлення вторгненнь

Існує велика кількість готових рішень IDS систем, але зосередимося на більш успішних сервісах від «великих» розробників – Snort, Suricata, Zeek(Bro), Cisco Firepower.

Характеристика	**Snort**	**Suricata**	**Zeek**	**Firepower**
Доступність	Безкоштовне	Безкоштовне	Безкоштовне	Підписка
Ефективність Виявлення	Висока ефективність в сигнатурному аналізі	Спроектований для швидкості та ефективності, протокольний аналіз	Спеціалізований на аналізі мережевого трафіку	Вбудовані механізми виявлення загроз, програмні + фізичні
Масштабованість	+	+	+	Великі корпоративні мережі
Тип аналізу	Сигнатурний аналіз	Сигнатурний так і аномальний аналіз	Аномальний аналіз	Сигнатурний так і аномальний аналіз

Характеристика	Snort	Suricata	Zeek	Firepower
Функціональність Журналів та Аналітика	Розширена	Розширена	Розширена	Розширена
Можливості Реагування на Інциденти	Виявлення	Виявлення	Виявлення	Виявлення + Блокування
Простота налаштування	Вимагає досвіду	Вимагає досвіду	Гарна документація	Вимагає досвіду

1.5. Формулювання завдань дослідження

У рамках цього дослідження передбачається вирішення ряду ключових завдань, спрямованих на створення ефективного рішення для виявлення та протидії мережевим атакам. Під час вибору області використання, вибір пав на кросбраузерний додаток, плагін, що матиме функціонал виявлення та захисту від загроз у мережі.

Для ефективної імплементації методів IDPS у веборієнтованих застосунках, необхідно ретельно спроектувати модель взаємодії. Функціональний підхід зосереджується на визначенні основних функцій та операцій, які браузер може виконувати в системі. Для цього можна

використовувати діаграми прецедентів та послідовності, що допоможе зрозуміти, як користувач буде взаємодіяти з додатком.

Важливою частиною моделювання є проєктування інтерфейсу, який дозволить користувачам комфортно та ефективно взаємодіяти з додатком. Дизайн інтерфейсу повинен бути інтуїтивно зрозумілим, а також забезпечувати можливість налаштування параметрів взаємодії під потреби користувачів.

Розробка системи містить процес проєктування архітектури, яка буде підтримувати взаємодію IDS як контролера з веборієнтованим додатком. Систему можна поділити на різні компоненти, які відповідають за різні аспекти взаємодії. Це можуть бути компоненти для зчитування даних з серверу який займається обробкою даних чи прикладні скрипти для розпізнавання загроз. Важливо ретельно розглянути структуру цих компонентів та їхню взаємодію.

Система повинна бути здатною інтегруватися з різними браузерами, мати свій сервер для обробки та зберігання даних, зручний інтерфейс відображення результатів аналітики користувачу та працювати в режимі real-time. Також має мати функціонал мануального блокування загроз, що потрібні користувачу.

2. МОДЕЛЮВАННЯ СИСТЕМИ

2.1. Поверхневе моделювання системи

Перед початком моделювання системи потрібно визначитися з основними вимогами та цілями дослідження. Основне завдання роботи це розробити програмний інструмент який може використовувати зовнішні сервери обробки даних виявлення мережевих атак. І в свою чергу матиме можливість блокувати загрози на основі аналітичної інформації, що він отримує. Також матиме гнучке налаштування функціоналу, адже не завжди аналітика IDPS може запобігти атаці без мануального втручання користувача. Перед тим як розробляти інструмент потрібно виявити як саме цей інструмент буде використовуватись.

З точки зору бізнесу сучасні IDPS системи звісно орієнтовані на користувача і розв'язування його проблем. Але найкращі представники свого класу мають недоліки зв’язані з підходом до рішення проблеми. Сьогодні, майже весь перелік взаємодії звичайного користувача з мережею складається з браузеру. Традиційні системи виявлення мережевих атак виявляють аномалії/сигнатури на моменті коли атака вже була прозведена, а далі намагаються усунути її. Тому найлогічнішим є рішення запобігання атаці, ще до того як вона була прозведена, та виявляти можливість небезпеки наперед. Найбільш ймовірне використання користувачем - це фонове сканування загроз при діях користувача та превентивна реакція при виявленні потенційної загрози. Звісно, додаток не може трактувати правила використання користувачу, тобто буде діяти більше як застереження. Іншим варіантом є використання кастомних фільтрів та правил, що дасть можливість користувачу заблокувати небезпечні сервіси заздалегідь. Також використання такої системи доцільно оцінювати в рамках користувача, який все ж таки не є професіоналом. Такому користувачеві потрібен простий спосіб захисту від зовнішніх чинників за способом «plug-and-play», тобто одразу після встановлення, без ніяких налаштуваннь. Способи

використання додатку представлено у вигляді діаграми прецедентів на рис. 2.1. З діаграми видно що основними акторами системи є користувач та саме веб-плагін. Очевидним прецедентом такої програми буде це довільне пересування, серфінг, мережею. Веб-плагін у свою чергу призначений для взаємодії з аналітичними функціями IDS серверу.

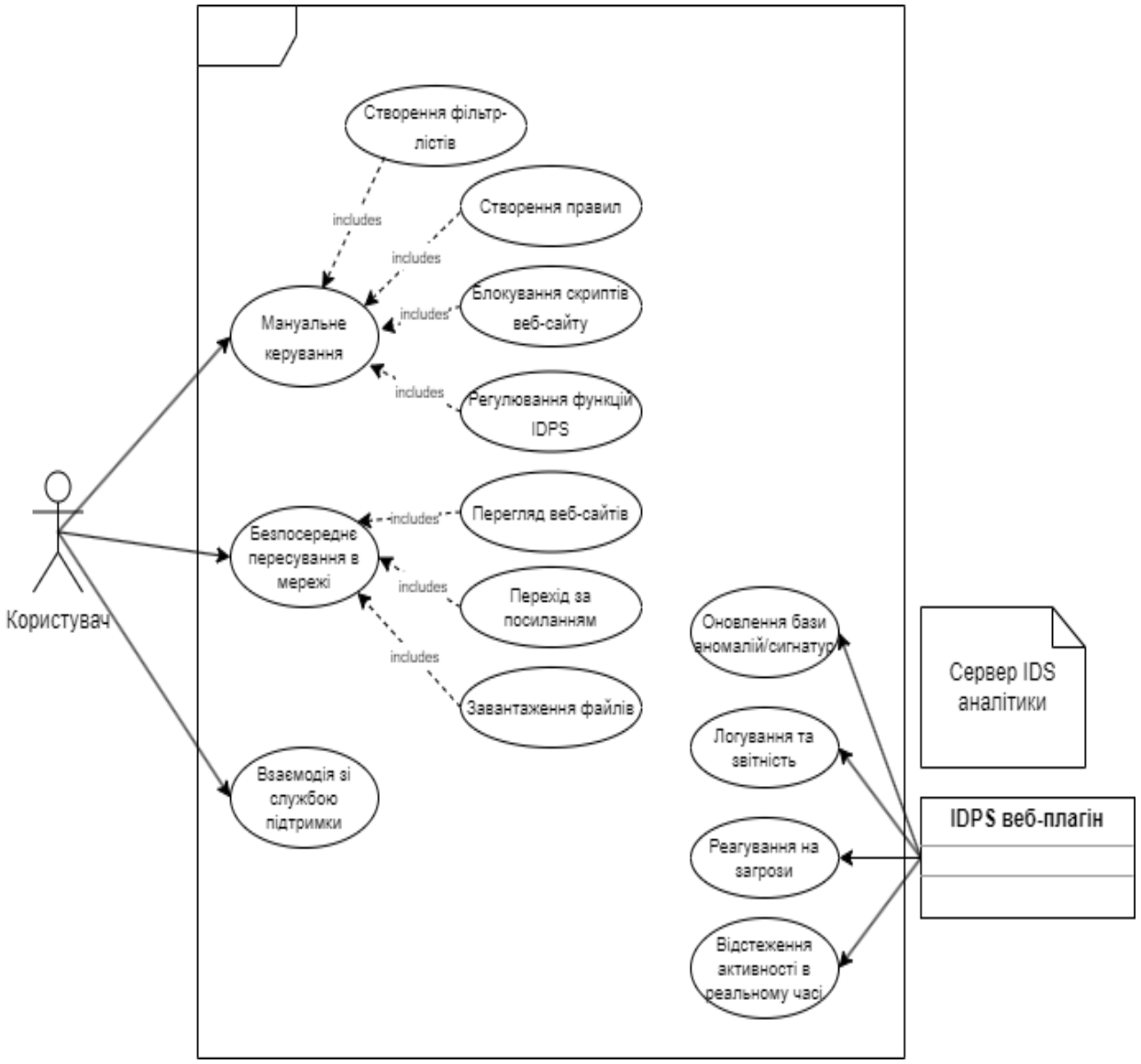

Рисунок 2.1 - діаграма прецедентів плагіну

Отже, виходячи з цієї діаграми прецедентів можна зробити висновки щодо структури майбутньої системи. У системі повинно бути явно відокремлено компонент, який займається обробкою даних пов’язаних з виявленням атак та превенції їх шкоди. Якщо поглянути на компонент який працює безпосередньо з виявленням та аналізом, то можна виділити такі прецеденти:

- Аналіз трафіку. Як мережевий так і сигнатурний аналізи для виявлення підозрілих патернів чи сигнатур вже відомих загроз
- Виявлення аномалій у мережевому трафіку, можливо навіть з використанням машинного навчання
- Журналювання та аналітика. Включає в себе запис подій для аналізу, історії виявлених атак, а також створення звітів та відображення статистики щодо виявлених загроз

Якщо розглянути компонент що займається запобіганню загрози можна виділити такі прецеденти:

- Автоматичне блокування, включає в себе безпосередньо блокування загрози та їх динамічне фільтрування
- Система налаштуваннь, що дозволяє налаштувати правила та фільтри для подальшого використання

Сам веб-плагін включає в себе:

- Користувацький інтерфейс, що є основним способом взаємодії. Попереджає про потенційні небезпеки і блокування атак. Також має можливості регулювання та налаштування систем.

- Ключовий функціонал – відстеження активності та взаємодія з сервером логіки IDPS, в режимі реального часу.

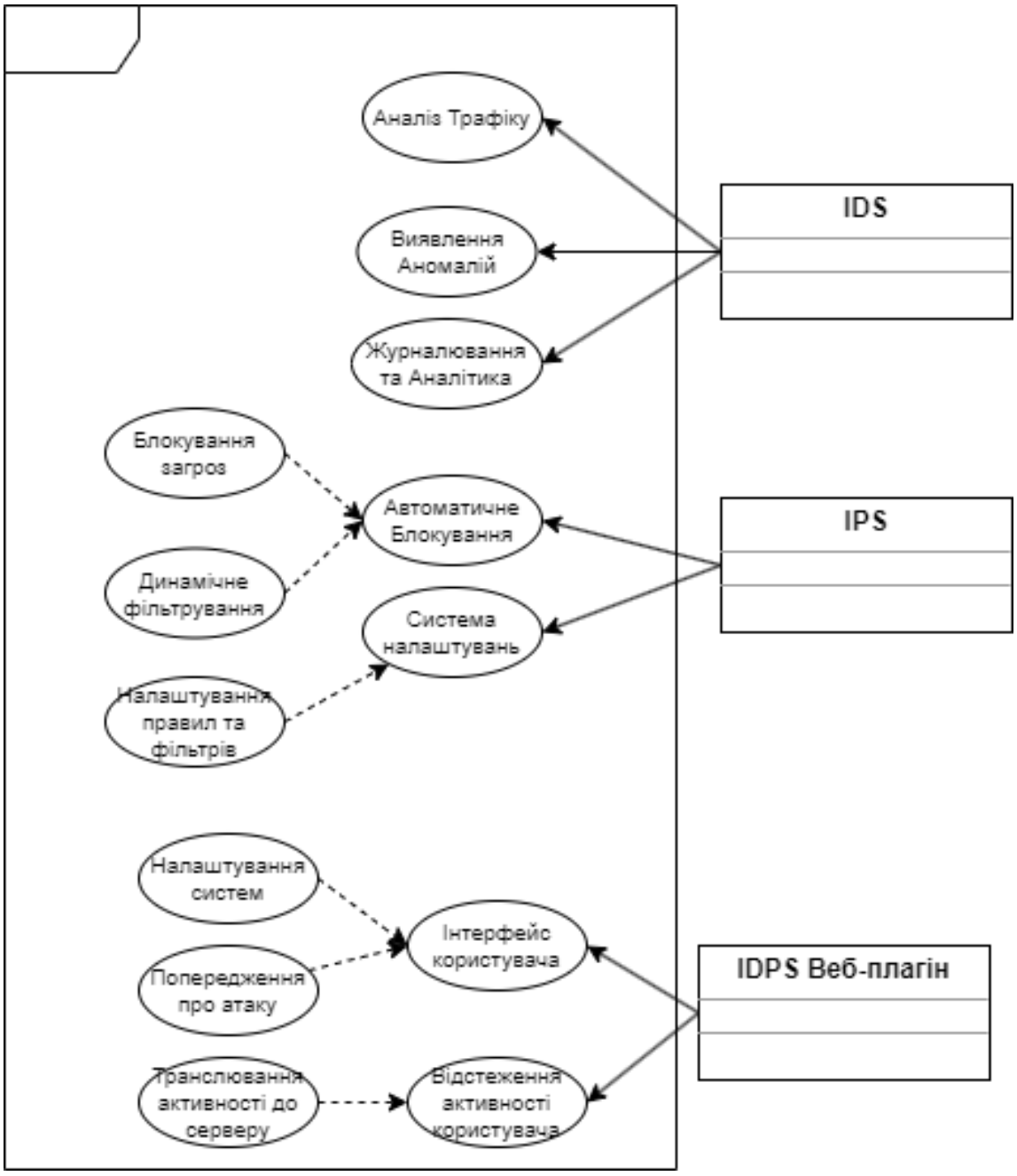

Рисунок 2.2 – Діаграма прецедентів IPDS системи

Детально взаємодію компонентів показано рис 2.3. у вигляді діаграми “послідовності”. Користувачеві потрібно встановити плагін до свого браузера та підключитися до бажаного серверу IDPS. Далі, при запуску браузера плагін буде виконувати з’єднання з аналітичною системою серверу та відстежувати дії користувача. При переході по незахищеній сторінці чи завантаженні, буде спрацьовувати тріггер на аналіз запиту. Якщо дія проходит перевірку на безпечність, плагін дозволить її виконати, інакше відбудеться блокування запиту і подальше сповіщення про причину блокування.

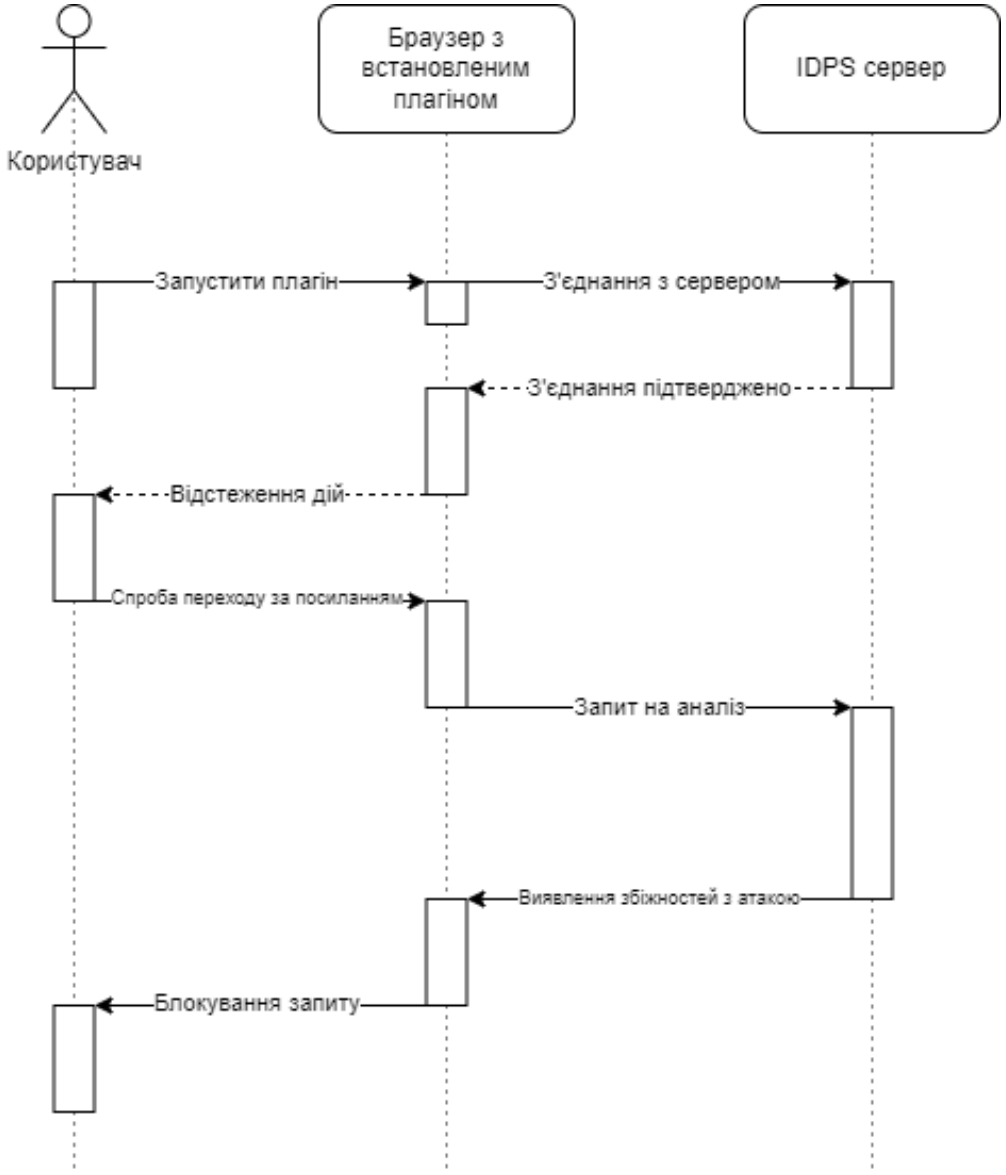

Рисунок 2.3. - діаграма послідовності процесу

Детальніше послідовність дій, які виконуються в рамках кожного прецеденту представлено на діаграмі активності. Отже, дослідивши можливе використання додатка користувачем можна виділити такі компоненти системи:

- Інтерфейс (UI) - вебзастосунок, в якому користувач може взаємодіяти з налаштуваннями та отримувати попередження про атаки
- SessionAnalyzer - компонент, який приймає/зчитує дії користувача для подальшого використання
- IDPS_API – API компонент, що відповідає за спілкування з сервером IDPS. Може бути взятий вже готовий варіант, чи написаний вручну на основі свого запровадженого серверу

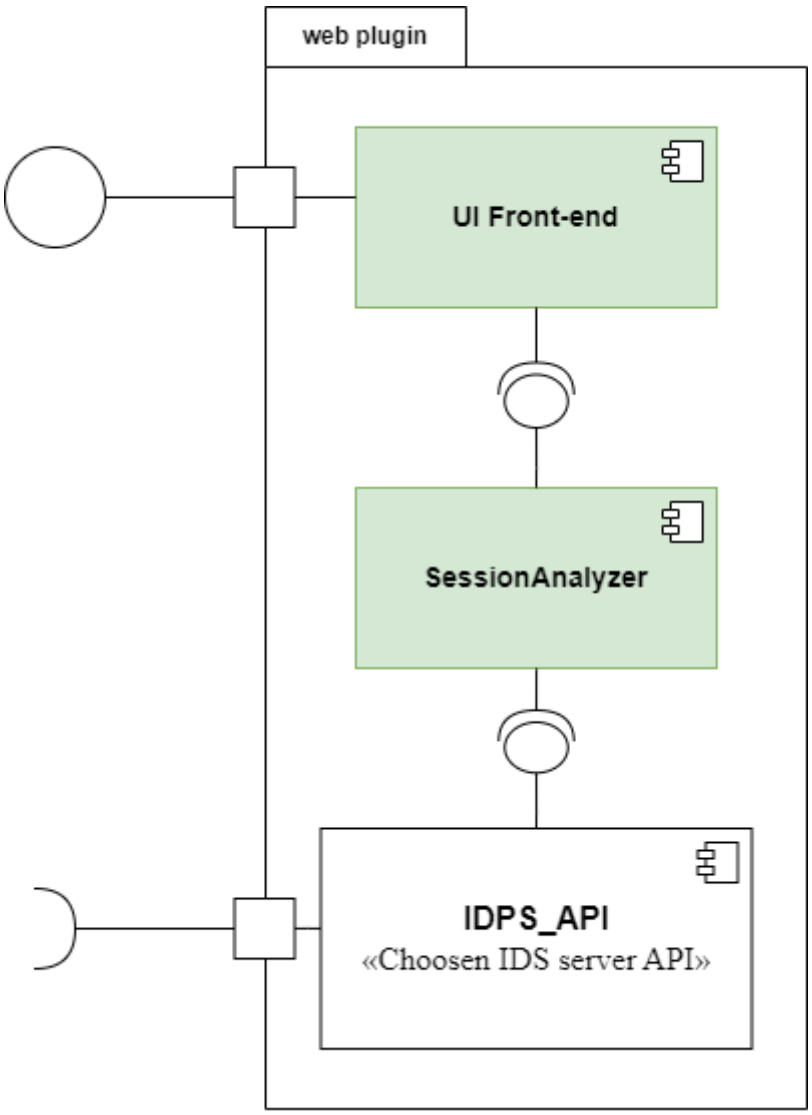

Рисунок 2.4 - діаграма компонентів

Основним компонентом що досліджується є компонент аналізу дій користувача, оскільки саме він відповідає за безперервний зв’язок з аналітичним сервером. Такий компонент представлений у вигляді бібліотеки на мові JavaScript - оскільки саме на цій мові можна розробляти веб-плагіни. Бібліотека повинна містити інструменти для блокування, транслювання та фільтрації дій користувача.

Взаємодію користувача з бібліотекою можна представити у вигляді діаграми активності, яка зображена на рисунку 2.6.

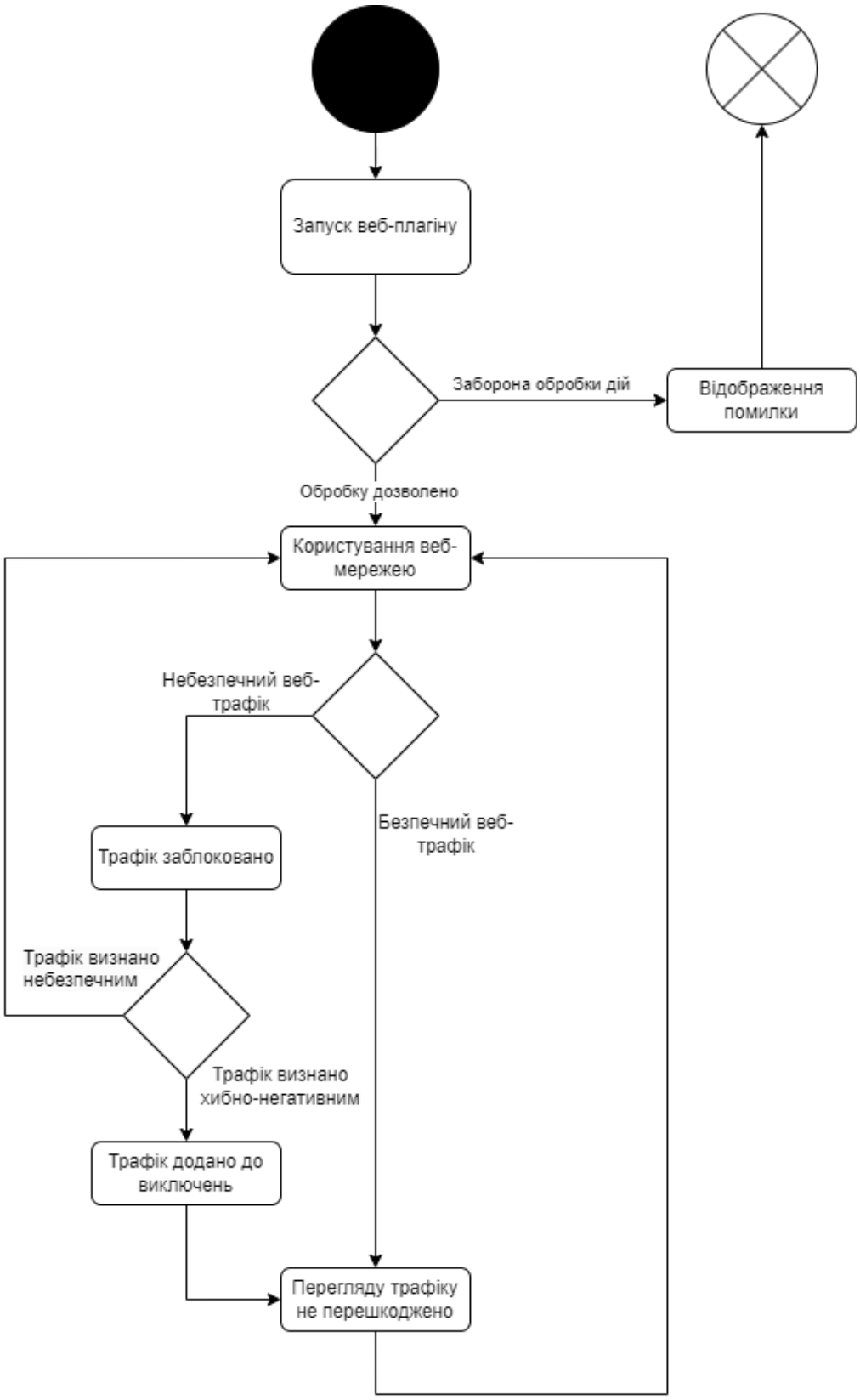

Рисунок 2.6. - діаграма активності роботи веборієнтованого додатка з використанням аналізу безпечності трафіку

2.2 Дослідження способів аналізу атак за допомогою штучного інтелекту

Складність кібератак ускладнює розробку ефективних інструментів для їх виявлення. Виявлення вторгнень на основі сигнатур було поширеним методом виявлення атак і забезпечення безпеки. Однак із появою штучного інтелекту, зокрема машинного навчання, глибокого навчання та ансамблевого навчання, були показані багатообіцяючі результати щодо більш ефективного виявлення атак.

Хмарні обчислення – це сучасна технологія, яка забезпечує ефективне зберігання та обробку даних. Через експоненціальне зростання даних, хмарні обчислення стали важливішими за традиційні обчислення в останні роки. Це широко поширена технологія через її масштабованість, економічну ефективність і гнучкість. Згідно з прогнозами Oracle, до 2025 року конфіденційних даних у хмарі буде в 600 разів більше.

Основною метою IDS є виявлення мережевих пакетів шляхом їх критичної перевірки та повідомлення про них адміністраторам шляхом генерації тривог. Після того, як традиційні технології виходять з ладу, IDS діє як адаптований захист для безпеки мережі. Оскільки хмарна інфраструктура обробляє величезний трафік, а традиційні технології не можуть виявити та класифікувати атаки, IDS може виявляти двійкові та багатокласові класифікації . Для двійкового коду модель виводить мітку як звичайні дані або дані атаки, а для мультикласу вихідні дані будуть багатозначними, що містять різні типи атак.

Основою системи виявлення є виявлення зловмисників і класифікація атак на основі ситуації. Окрім виявлення вторгнень, також можуть бути передбачені можливості запобігання можливим інцидентам. Комбінований механізм називається системами виявлення та запобігання вторгненням. Залежно від

методу виявлення розрізняють механізми виявлення на основі сигнатур і на основі аномалій. Механізми виявлення вторгнень на основі сигнатур ідентифікують конкретні шаблони та порівнюють шаблон із лише спостережуваними подіями. Хоча метод може виявити всі відомі атаки, оскільки шаблони відрізняються для виявлення нових атак, цього механізму недостатньо. Методи IDS також можуть виявляти аномалії, виявляючи будь-які відхилення від нормальної поведінки. Цей механізм також називають виявленням неправильного використання.

Механізм виявлення базується виключно на трьох елементах: параметризації, навчанні та виявленні. На кроці параметризації представлено спостережувану поведінку, наприклад мережеві підключення та хости. Щоб побудувати класифікаційну модель, яка класифікує спостережувану поведінку як нормальну або ненормальну, використовується навчальний модуль. Для виявлення класифікована модель, побудована на попередньому кроці, використовується для прогнозування нових аномалій.

Однак більшість досліджень базуються на згаданих вище традиційних моделях, і жоден із методів не є потужним. Проблему можна сформулювати з точки зору машинного навчання(Machine Learning), а також з точки зору глибинного навчання(Deep Learning), яка є підгалуззю ML. Результатом рішення ML є мітка з normal або attack. Незважаючи на те, що попередні моделі ML показали продуктивність порівняно з традиційними алгоритмами, жодна з них не продемонструвала чудової продуктивності, показавши високу точність і низьку частоту помилкових тривог. Згідно з попередніми дослідженнями, метод DL виявлення вторгнень є вигідним для прогнозування атак з високою точністю. Іншими словами, система виявлення вторгнень на основі DL показує кращий прогноз, ніж ML.

Узагальнивши відповідні дослідження з точки зору класифікації щодо ML, DL та систем виявлення вторгнень на основі ансамблю. Класифікація складається в основному з продуктивності та алгоритмів. Класифікувавши пов’язані дослідження з точки зору ML, DL та систем виявлення вторгнень на основі ансамблю та порівнявши продуктивність із показниками продуктивності.

У цьому розділі можна побачити класифікацію систем виявлення вторгнень на основі ШІ. На рис. 2.7. детально представлені методи виявлення вторгнень на основі ШІ з точки зору ML, DL і ансамблевого навчання. Новизна класифікації полягає в тому, щоб зібрати всі типи методологій навчання ML, DL та ансамблю, які мають можливість виявляти зловмисників.

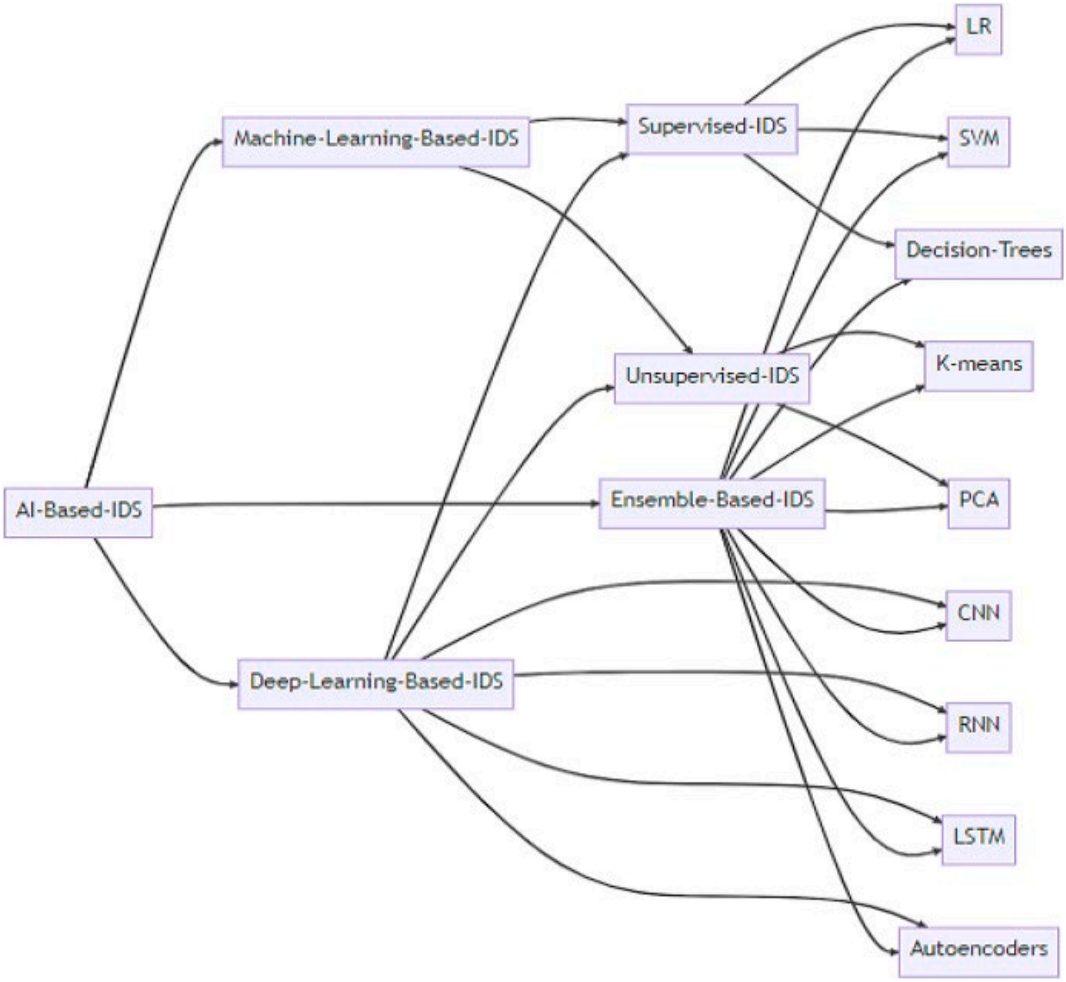

Рисунок 2.7. - класифікація IDS на основі ШІ

3. РОЗРОБКА АЛГОРИТМІВ IDPS ВЕБ-СИСТЕМИ

3.1. Вибір інструментів для розробки

Для написання веб-плагінів зазвичай використовують популярні мови програмування та фреймворки. JavaScript є основною мовою для розробки клієнтської частини плагінів, оскільки вона підтримується всіма сучасними браузерами. Але краще використовувати мову програмування TypeScript, яка може компілюватися у мову JavaScript. Мова TypeScript надає можливість визначати типи даних для змінних, функцій та об'єктів, що допомагає виявляти помилки під час розробки на етапі компіляції, а не під час виконання. Це сприяє створенню більш надійних та стабільних бібліотек. Також строга типізація дає підказки користувачам, які використовують бібліотеку, адже вони завжди знають що роблять конкретні функції чи класи бібліотеки. Оскільки веб-плагін має використовуватися у браузері, його обов’язково потрібно перетворити у JavaScript. Для цього використовується система збірки Vite. Vite - це надбудова над системами збірки EsBuild та Rollup. EsBuild швидко компілює TypeScript до JavaScript, а Rollup керує процесом збірки бібліотеки.

Основні інструменти включають Chrome Extension API для Google Chrome, WebExtension API для браузерів, таких як Mozilla Firefox, і Microsoft Edge Extension API для Edge. Розширені фреймворки, такі як React або Vue.js, можуть використовуватися для полегшення розробки інтерфейсу користувача. Для моніторингу та фільтрації HTTP-запитів можна використовувати бібліотеки, такі як Axios. У разі необхідності захисту від малвар та інтеграції з антивірусними движками використовують відповідні технології. Важливо також дотримуватися принципів безпеки та використовувати шифрування для захисту конфіденційної інформації.

Також нам потрібно буде побудувати та налаштувати IDPS сервер обробки даних. Для побудови сервера, нам може бути корисно використовувати різноманітні інструменти та технології. Snort є одним з популярних відкритих інструментів для IDS, але є дуже вдалий та сучасний варіант з відкритим кодом Zeek. Має на порядок кращі налаштування та документацію. Він дозволяє виявляти потенційні інтрузії, аналізуючи мережевий трафік.

Для побудови IDS бекенд-серверу вибрано мову програмування Python, адже має дуже простий спосіб реалізації ідей. Для аналітичної частини використовуємо бібліотеку Zeek у поєднанні з фреймворком Flask.

Спершу, встановлюємо Zeek для аналізу мережевого трафіку та виявлення інтрузій. Далі, використовуємо Flask для створення RESTful API, яке буде обробляти дані, зібрані Zeek. Flask дозволяє легко створювати веб-сервер та реалізовувати взаємодію з іншими компонентами, такими як веб-плагін.

Загальна структура може включати Zeek як компонент виявлення інтрузій, Flask для створення API, а також додаткові бібліотеки Python для обробки та аналізу даних. Такий підхід дозволяє забезпечити потужний та гнучкий IDS сервер на основі мови програмування Python.

Для збору та аналізу журналів подій, ELK (Elasticsearch, Logstash, Kibana) стане корисним стеком. Elasticsearch використовується для зберігання та пошуку данних, Logstash для централізованої обробки журналів, а Kibana для візуалізації та аналізу даних.

3.2. Архітектура бібліотеки

Систему можна поділити на 3 модулі:

- Web Plugin — виконує аналіз трафіку та дій браузера. Взаємодіє з сервером.

- Web Browser – має функціонал що потребує трекінгу, також є гніздом для монтування веб-плагіну
- IDPS Server — виконує роль «мозоку», складні логічні та аналітичні процеси, обробку даних та їх зберігання

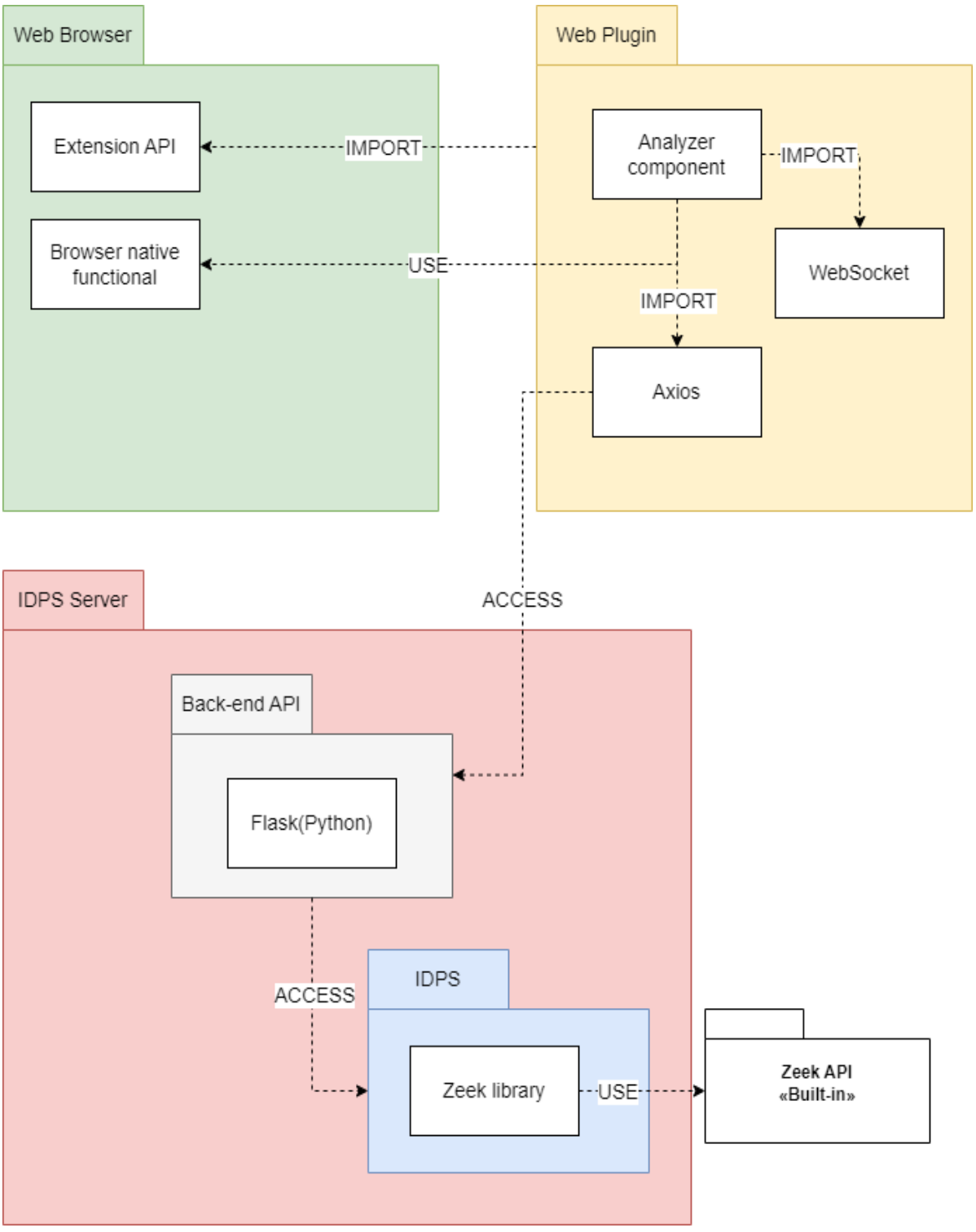

Рисунок 3.1 - діаграма пакетів

Модуль Web Browser складається з таких частин:

- Extension API – пакет, що дає можливість використовувати веб-плагіни, та ссилатися на нативний функціонал браузеру.
- Browser Native Functional – основний модуль браузеру, який потрібно відстежувати модулем аналізу веб-плагіну.

Сам модуль плагіну Web Plugin включає такі частини:

- Analyzer – модуль, що користується Extension API щоб мати доступ до відстеження основних функцій браузера. Тобто, трекінг трафіку, блокування трафіку тощо.
- Axios – просунутий пакет для HTTP взаємодії з сервером обробки інформації та його API.
- WebSocket – пакет для створення зв’язку типу реального часу з сервером IDPS.

Модуль аналізу створює зв’язок з нативним функціоналом браузеру шляхом прив’язки до евентів та створенням власної карти реакцій на кожен, з подальшою взаємодією з API серверу IDPS.

Модуль IDPS серверу складається з:

- IDPS – модуль головної системи логіки та аналітики, використовує алгоритми бібліотеки Zeek.
- Back-end API – модуль методів бекенду, написаний за допомогою Flask (Python). Є прошарком взаємодії між інтерфейсом користувача та самою IDPS системою, шляхом зв’язку підготовлених ендпоїнтів з методами Zeek бібліотеки.

3.3. Принцип роботи трекера дій

Браузерний інтерфейс WebAPI надає інструментарій для отримання повного контролю над браузером та його діями. Вміст такого масиву функцій представлено на рисунку 3.2.

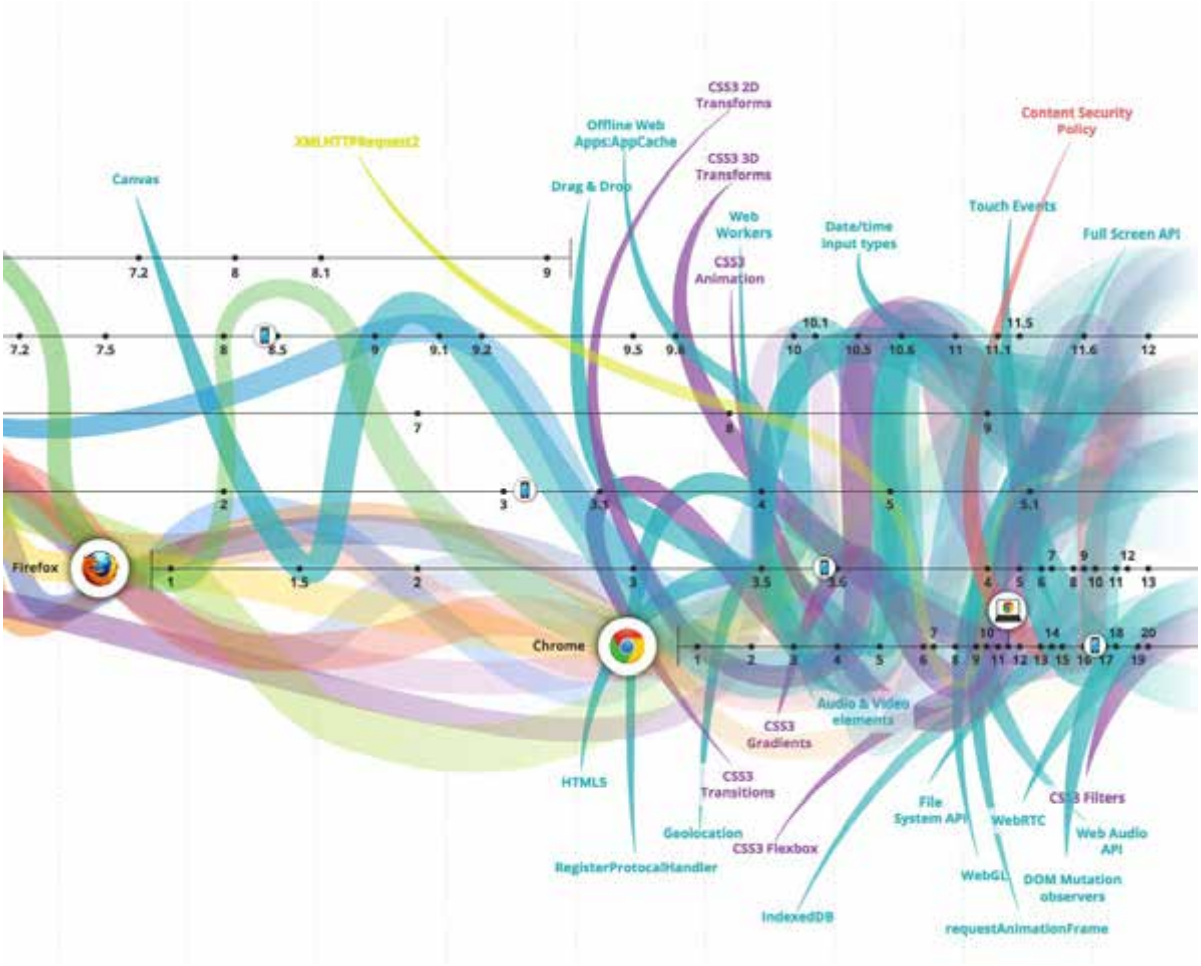

Рисунок 3.2 - вміст функцій браузеру для відстеження

Алгоритм відстеження дій в браузері може бути реалізований через використання слухачів подій (event listeners) в мові програмування JavaScript. Спочатку, визначте необхідні події для відстеження, такі як кліки, наведення

миші, введення тексту тощо. Потім встановіть відповідні слухачі подій для елементів, на які ви хочете відстежувати дії.

При кліці чи іншій взаємодії користувача, слухач подій викличе відповідну функцію. У цій функції ви можете здійснювати різні дії, такі як збір та відправка даних на сервер для подальшого аналізу чи збереження локально.

Наприклад, для відстеження кастомних евентів ви можете використовувати наступний код:

main.js

```
// content.js (код внутрішньої частини веб-плагіна)

// Функція для логування подій
function logEvent(eventName, details) {
    console.log(`Подія "${eventName}" відстежена. Деталі:`, details);
}

// Додавання слухача подій для кліків на будь-якому елементі сторінки
document.addEventListener('click', function(event) {
    logEvent('click', { target: event.target });
});

// Додавання слухача подій для введення тексту в будь-якому полі вводу
document.addEventListener('input', function(event) {
    logEvent('input', { target: event.target, value: event.target.value });
});

// Додавання слухача подій для інших подій, якщо потрібно
document.addEventListener('your_custom_event', function(event) {
    logEvent('your_custom_event', { customDetail: 'your_custom_detail' });
});

// Ваш плагін може виконувати інші дії відповідно до потреб вашого проекту

```

Рисунок 3.3 – функціонал трекінгу дій користувача

Приклад впровадження веб-сокету що буде підтримувати постійний зв’язок з сервером 3.4.

```
// Встановлення WebSocket-з'єднання
const socket = new WebSocket('ws://your-ids-server-url');

// Обробник подій при відкритті з'єднання
socket.addEventListener('open', (event) => {
  console.log('WebSocket з\'єднання відкрито:', event);
});

// Обробник подій при отриманні повідомлення від сервера IDS
socket.addEventListener('message', (event) => {
  console.log('Повідомлення від сервера IDS:', event.data);
});

// Обробник подій при закритті з'єднання
socket.addEventListener('close', (event) => {
  console.log('WebSocket з\'єднання закрито:', event);
});

// Відправка даних на сервер IDS при виникненні події відстеження
function trackUserAction(actionDetails) {
  const message = JSON.stringify({ action: 'track', details: actionDetails });
  socket.send(message);
}
```

Рисунок 3.4 – Веб-сокет на стороні веб-плагіну

```python
from flask import Flask, render_template
from flask_socketio import SocketIO

app = Flask(__name__)
socketio = SocketIO(app)

@socketio.on('connect')
def handle_connect():
    print('Клієнт підключений')

@socketio.on('message')
def handle_message(message):
    print('Повідомлення від клієнта:', message)

    # Тут ви можете обробляти отримані дані та реагувати відповідно до потреб IDS

@socketio.on('disconnect')
def handle_disconnect():
    print('Клієнт відключений')

if __name__ == '__main__':
    socketio.run(app, debug=True)
```

Рисунок 3.4 – Веб-сокет на стороні серверу

Функціонал Axios частини для взаємодії з сервером IDPS:

```
// Встановлення Axios (переконайтеся, що ви підключили бібліотеку)
const axios = require('axios');

// URL сервера IDPS API
const idpsApiUrl = 'https://your-idps-server/api';

// Функція для відправки GET-запиту на сервер IDPS
async function fetchDataFromIDPS(endpoint, params = {}) {
    try {
        const response = await axios.get(`${idpsApiUrl}/${endpoint}`, { params })
        return response.data;
    } catch (error) {
        console.error('Помилка під час виконання GET-запиту:', error.message);
        throw error;
    }
}

// Функція для відправки POST-запиту на сервер IDPS
async function sendDataToIDPS(endpoint, data) {
    try {
        const response = await axios.post(`${idpsApiUrl}/${endpoint}`, data);
        return response.data;
    } catch (error) {
        console.error('Помилка під час виконання POST-запиту:', error.message);
        throw error;
    }
}

// Приклад використання:

// Запит для отримання списку інцидентів
const incidents = fetchDataFromIDPS('incidents');
console.log('Отримані інциденти:', incidents);

// Припустимо, що у вас є дані, які ви хочете відправити на сервер IDPS
const newData = { action: 'some_action', details: 'some_details' };

// Відправка даних на сервер IDPS
const result = sendDataToIDPS('log', newData);
console.log('Результат відправлення даних:', result);


```

Рисунок 3.5 - алгоритм Axios частини

Реалізація класу трекінгу за допомогою плагіну:

```
// Analyzer компонент
class UserActionAnalyzer {
    constructor() {
      this.userActions = [];
      this.initialize();
    }

    // Ініціалізація обробників подій
    initialize() {
      this.attachEventListeners();
      this.trackNavigation();
    }

    // Додавання обробників подій для відстеження різних дій користувача
    attachEventListeners() {
      document.addEventListener('click', this.handleUserClick.bind(this));
      document.addEventListener('keydown', this.handleKeyPress.bind(this));
      window.addEventListener('scroll', this.handleScroll.bind(this));
      // Додайте інші обробники подій за потреби
    }

```

Рисунок 3.6 – початок реалізації класу UserActionAnalyzer

```
    // Обробка кліків користувача
    handleUserClick(event) {
      const action = {
        type: 'click',
        target: event.target.tagName,
        timestamp: new Date().toISOString(),
      };
      this.userActions.push(action);
      this.sendDataToServer(action);
    }

    // Обробка натискання клавіш користувачем
    handleKeyPress(event) {
      const action = {
        type: 'keypress',
        key: event.key,
        timestamp: new Date().toISOString(),
      };
      this.userActions.push(action);
      this.sendDataToServer(action);
    }

    // Обробка прокрутки користувачем
    handleScroll(event) {
      const action = {
        type: 'scroll',
        scrollTop: window.scrollY,
        timestamp: new Date().toISOString(),
      };
      this.userActions.push(action);
      this.sendDataToServer(action);
    }

    // Відстеження навігації
    trackNavigation() {
      window.addEventListener('popstate', () => {
        const action = {
          type: 'navigation',
          location: window.location.href,
          timestamp: new Date().toISOString(),
        };
        this.userActions.push(action);
        this.sendDataToServer(action);
      });
    }

    // Відправка даних на сервер аналізу
    sendDataToServer(data) {
      // Логіка для відправки даних на сервер, наприклад, через AJAX або WebSocket
      console.log('Відправлено на сервер:', data);
    }
  }

  // Створення екземпляру аналізатора
  const userAnalyzer = new UserActionAnalyzer();
```

Рисунок 3.7 – завершення реалізації класу UserActionAnalyzer

Функціонал взаємодії класу трекінгу з класом відправки на сервер

```
class ServerInteraction {
    static sendData(userActions) {
      // Відправка даних на сервер аналізу
      return axios.post('https://your-analytics-server/api/data', { userActions })
        .then(response => {
          console.log('Відповідь від сервера:', response.data);
        })
        .catch(error => {
          console.error('Помилка під час відправки даних:', error);
          throw error;
        });
    }

    static getAnalyticsData(userId) {
      // Отримання аналітичних даних для конкретного користувача
      return axios.get(`https://your-analytics-server/api/analytics?userId=${userId}`)
        .then(response => {
          console.log('Аналітичні дані для користувача:', response.data);
        })
        .catch(error => {
          console.error('Помилка під час отримання аналітичних даних:', error);
          throw error;
        });
    }
  }

  // Розширення класу UserActionAnalyzer для використання ServerInteraction
  class ExtendedUserActionAnalyzer extends UserActionAnalyzer {
    constructor(userId) {
      super();
      this.userId = userId;
    }

    // Перевизначення функції відправки даних на сервер
    sendDataToServer(data) {
      // Додаткова обробка або маніпуляції з даними перед відправкою, якщо потрібно
      super.sendDataToServer(data);

      // Відправка даних на сервер аналізу за допомогою Axios
      ServerInteraction.sendData(data);
    }

    // Функція отримання аналітичних даних для поточного користувача
    getAnalyticsData() {
      ServerInteraction.getAnalyticsData(this.userId);
    }
  }

  // Створення екземпляру розширеного аналізатора з ідентифікатором користувача
  const extendedUserAnalyzer = new ExtendedUserActionAnalyzer('123');

  // Відстеження дій користувача
  // ... (код для виклику методів аналізатора, наприклад, extendedUserAnalyzer.handleUserClick())
```

Рисунок 3.7 – завершення реалізації класу ServerInteraction

Оскільки алгоритми IDPS серверу знаходятся у бібліотеці, можна переглянути алгоритми фільтрації трафіку Zeek у спрощеному вигляді:

```cpp
#include <iostream>
#include <string>
#include <unordered_map>

// Структура для представлення мережевого пакету
struct NetworkPacket {
    std::string sourceIP;
    std::string destinationIP;
    uint16_t sourcePort;
    uint16_t destinationPort;
    std::string protocol;
    std::string payload;
};

// Клас для аналізу мережевих подій
class ZeekAnalyzer {
public:
    // Конструктор
    ZeekAnalyzer() {
        std::cout << "Zeek Analyzer ініціалізовано" << std::endl;
    }

    // Метод для аналізу HTTP-трафіку
    void analyzeHTTP(const NetworkPacket& packet) {
        // Ваш аналіз HTTP-запитів тут
        std::cout << "HTTP Request: " << packet.sourceIP << ":" << packet.sourcePort
                  << " -> " << packet.destinationIP << ":" << packet.destinationPort
                  << " | URI: " << extractURI(packet.payload) << std::endl;
    }

    // Метод для аналізу DNS-трафіку
    void analyzeDNS(const NetworkPacket& packet) {
        // Ваш аналіз DNS-запитів тут
        std::cout << "DNS Request: " << packet.sourceIP << ":" << packet.sourcePort
                  << " -> " << packet.destinationIP << ":" << packet.destinationPort
                  << " | Query: " << extractDNSQuery(packet.payload) << std::endl;
    }

private:
    // Метод для видобування URI з HTTP-пакету
    std::string extractURI(const std::string& payload) {
        // Логіка видобування URI тут
        return "example_uri";
    }

    // Метод для видобування DNS-запиту з DNS-пакету
    std::string extractDNSQuery(const std::string& payload) {
        // Логіка видобування DNS-запиту тут
        return "example_query";
    }
};

int main() {
    // Створення об'єкта аналізатора
    ZeekAnalyzer zeekAnalyzer;

    // Приклад аналізу HTTP-трафіку
    NetworkPacket httpPacket = {"192.168.1.1", "8.8.8.8", 1234, 80, "HTTP", "GET /index.html"};
    zeekAnalyzer.analyzeHTTP(httpPacket);

    // Приклад аналізу DNS-трафіку
    NetworkPacket dnsPacket = {"192.168.1.2", "8.8.8.8", 5678, 53, "DNS", "example.com"};
    zeekAnalyzer.analyzeDNS(dnsPacket);

    return 0;
}
```

Рисунок 3.8 – спрощений вигляд алгоритмів Zeek

Приклад роботи скриптів та аналізаторів у бібліотеці Zeek. Написані кастомною мовою BroScript.

```
module HTTP;

event http_request(c: connection, method: string, uri: string)
    {
    print(fmt("HTTP Request: %s %s", method, uri));
    }

event http_reply(c: connection, status_code: count)
    {
    print(fmt("HTTP Reply: %s", status_code));
    }

@load HTTP

event zeek_init()
    {
    print("Zeek ініціалізовано");
    }

event http_packet(c: connection, is_orig: bool, method: string, uri: string, status_code: count)
    {
    if (is_orig)
        {
        HTTP::http_request(c, method, uri);
        }
    else
        {
        HTTP::http_reply(c, status_code);
        }
    }

- http_analyzer.zeek
- main.zeek

zeek -Cr your_network_traffic.pcap main.zeek

```

Рисунок 3.9 – скрипти і аналізатори Zeek

Для розробки ендпоїнтів за допомогою Flask розроблено такий патерн:

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

# Приклад даних, які можуть бути використані сервером Zeek
zeek_data = {"status": "OK", "message": "Zeek server is running."}

# Роут для отримання інформації від сервера Zeek
@app.route('/zeek_info', methods=['GET'])
def get_zeek_info():
    return jsonify(zeek_data)

# Роут для отримання даних від веб-плагіну та їх передачі серверу Zeek
@app.route('/send_data_to_zeek', methods=['POST'])
def send_data_to_zeek():
    if not request.json or 'data' not in request.json:
        return jsonify({'error': 'Invalid data format'}), 400

    data_from_plugin = request.json['data']

    # Тут ви можете виконати обробку даних та передати їх серверу Zeek
    # Замість print використовуйте логіку взаємодії з сервером Zeek
    print(f"Received data from plugin: {data_from_plugin}")

    # Повертання відповіді
    return jsonify({'status': 'Data received successfully'}), 200

if __name__ == '__main__':
    app.run(debug=True)

```

Рисунок 3.10 – приклад ендпоїнтів що використовуватиме веб-плагін

3.5. Вибір інструментів для розробки

Очевидно, що для того, щоб розробляти продукти для веббраузера необхідно використовувати мови HTML, CSS та JavaScript. Однак, для побудови бібліотек краще використовувати мову TypeSciprt замість JavaScript. Мова TypeScript надає додатковий синтаксис для опису типів, які використовуються у коді та компілюється у JavaScript. Типи дають підказки у середовищі розробки та надають можливість показати помилку при неправильному використанні класів та функцій з бібліотеки. Також, під час розробки веб-плагіну можна використовувати фреймворк VueJS, для створення компонента відтворення звуку. VueJS - це прогресивний фреймворк, який має власну систему реактивності, рендерінгу та можливість компіляції у Web Components - стандарт для браузера, який дозволяє створювати використовувати власні HTML елементи. Оскільки архітектура VueJS опирається на композицію компонентів, бібліотека експортує спеціальні функції (composables), якими можна розширяти інші компоненти. Для простого способу написання таких функцій використовується бібліотека з готовими інструментами vueuse.

Також необхідно використати Extensions API який дозволяє перетворити ваш веб-додаток у плагін. Для швидкого написання стилів використовувався інструмент unocss, який значно спрощує написання CSS коду надаючи готові стильові рішення.

Для збірки бібліотеки використовується інструмент збірки проєктів vite. Для написання тестів використовується бібліотека vitest.

Для написання коду ккраще використати IDE Visual Studio Code.

3.6. Практична реалізація розробленого методу

Для тестування та демонстрації можливостей IDPS, було створено веб-плагін.

Плагін має наступні частини:

- Для побудови вебзастосунку використовується вебфреймворк VueJS та ExtensionAPI. Інтерфейс поділений на такі частини:
- Інформаційна — поточна сторінка, її статус;
- Налаштування — окрема сторінка з налаштуваннями плагіну та самого IDPS серверу;
- Логгер — інструменти для відображення та створення логів;
- Підтримка – інструмент для зв’язку зі службою підтримки;
- Аналітика – що було заблоковано і коли;
- Фільтрація – статус бар з функціями блокування та фільтрації трафіку.

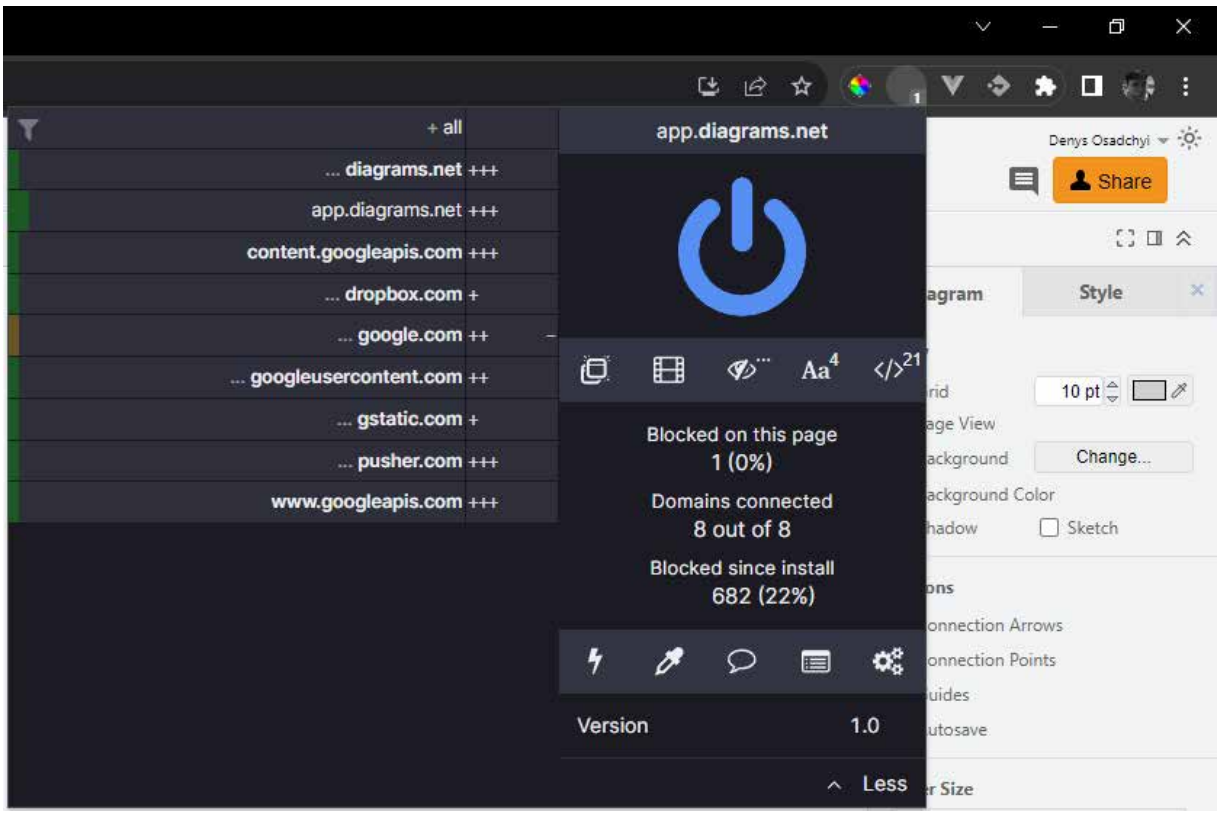

Рисунок 3.6 – інтерфейс плагіну

Для сторінки з налаштуваннями приспособлено великий обсяг функціоналу IDPS серверу:

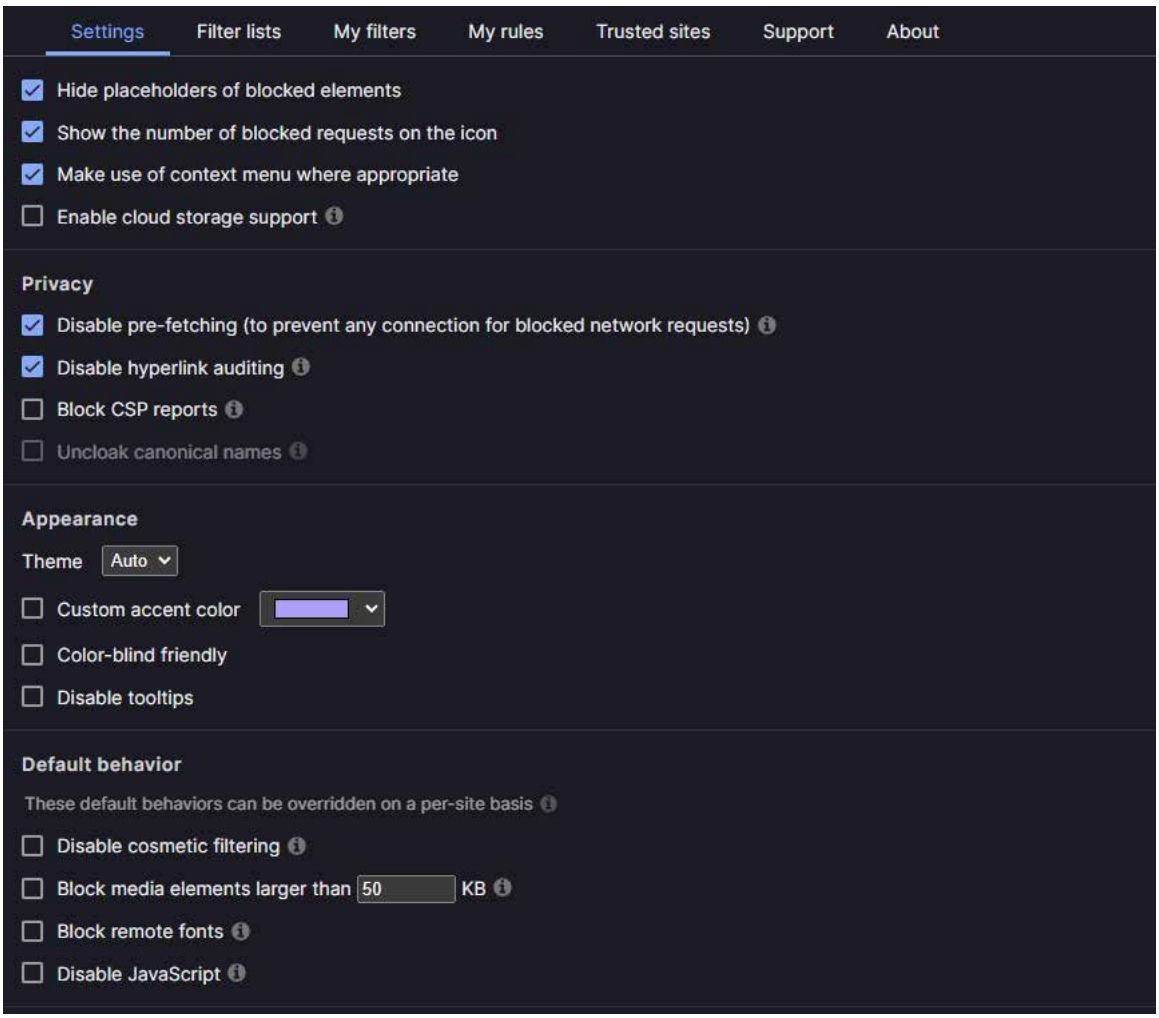

Рисунок 3.17 – налаштування плагіну

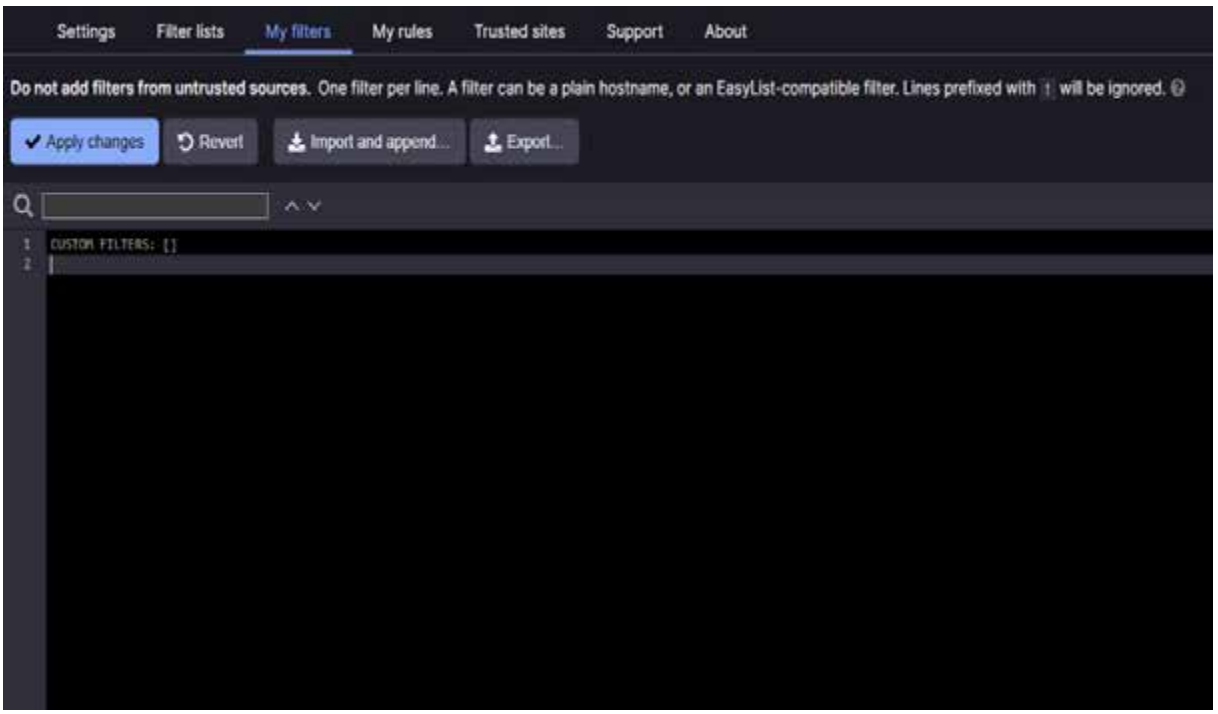

Рисунок 3.18 – налаштування фільтрів серверу

Налаштування правил серверу:

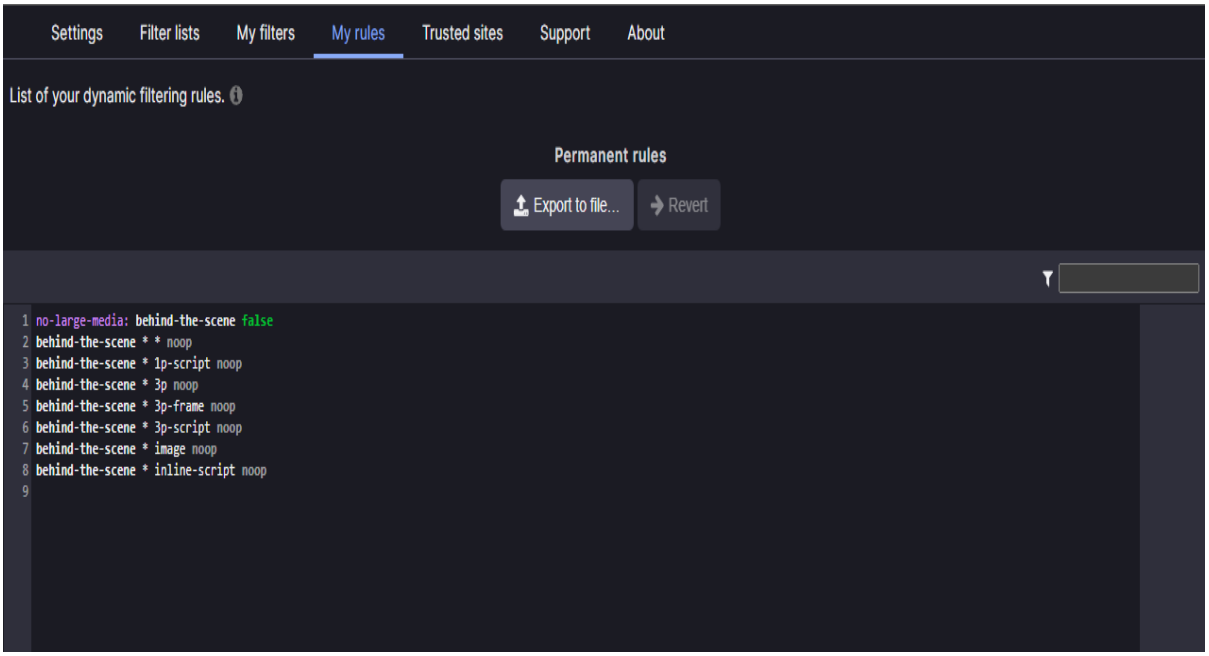

Рисунок 3.19.1 – налаштування правил серверу

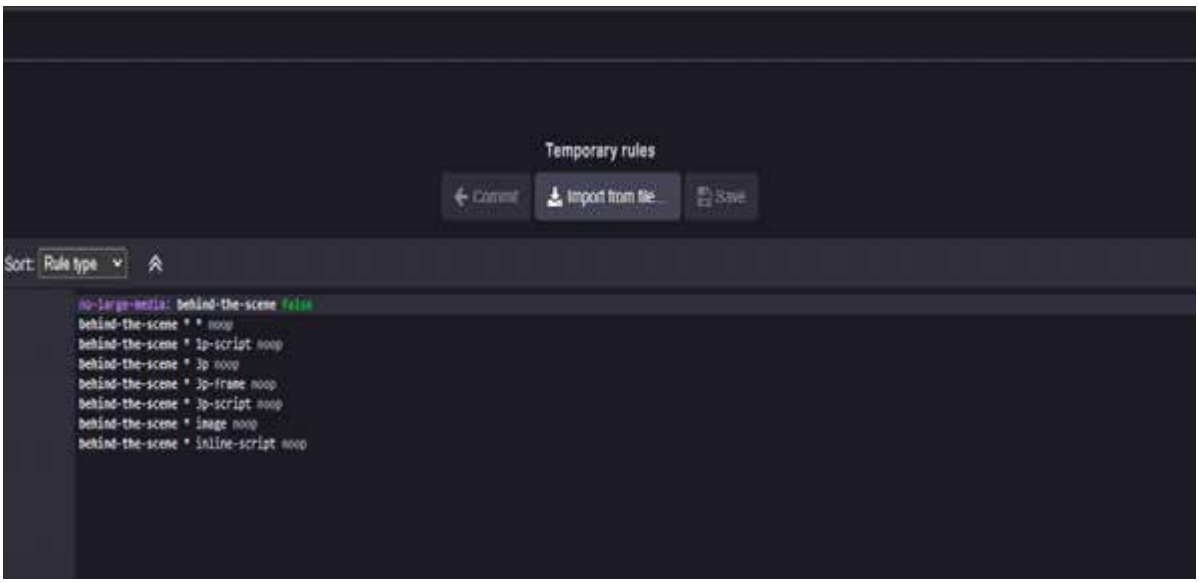

Рисунок 3.19.2 – налаштування правил серверу

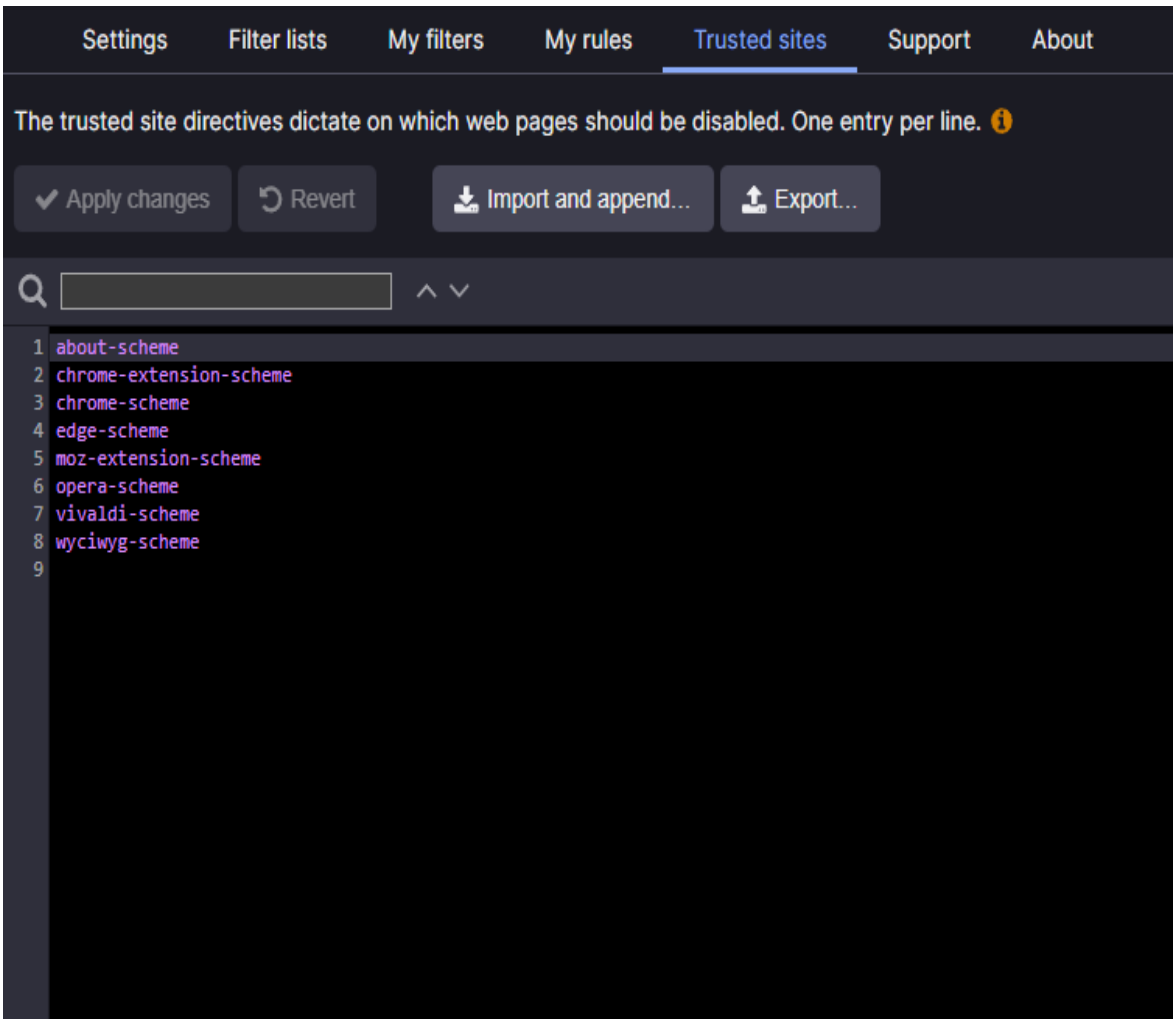

Рисунок 3.20 – налаштування виключень серверу

4. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

4.1. Потенційна сфера застосування

Веб-плагін, спрямований на захист у веб-мережі з використанням системи виявлення та запобігання вторгнень (IDPS), може знайти широке застосування в різних областях та принести значні переваги в плані кібербезпеки.

Однією з потенційних сфер застосування є бізнес-сегменти. Великі корпорації та підприємства можуть використовувати веб-плагін для захисту своїх веб-систем від різноманітних кіберзагроз, включаючи SQL-ін'єкції, кросс-сайт скриптінг та інші атаки. Це дозволяє забезпечити надійний захист конфіденційної інформації та уникнути витоків даних.

У сфері e-commerce веб-плагін може бути використаний для захисту інтернет-магазинів від атак, спрямованих на клієнтські дані та фінансову інформацію. Це сприяє підвищенню довіри покупців та забезпеченню безпечних та надійних транзакцій.

У сфері організаційного управління веб-плагін може бути використаний для захисту внутрішніх корпоративних систем від зовнішніх атак та неправомірного доступу. Це особливо важливо для організацій, які обробляють конфіденційні дані клієнтів чи великі обсяги корпоративної інформації.

В галузі охорони здоров'я важливо забезпечити безпеку медичних систем та збереження конфіденційності пацієнтської інформації. Веб-плагін, який інтегрується з IDPS, може допомогти у запобіганні атак, спрямованих на медичні бази даних та інформаційні системи лікарень та клінік.

У освітньому секторі веб-плагін може забезпечити захист для веб-порталів університетів та шкіл, забезпечуючи конфіденційність особистих даних студентів та вчителів.

Загалом, веб-плагін для захисту у веб-мережі, що базується на IDPS, має широкий спектр застосування та є ключовим інструментом для забезпечення безпеки в онлайн-середовищі. Він допомагає уникнути ризиків вторгнень, збільшити впевненість користувачів та забезпечити стійкість веб-систем до різноманітних кіберзагроз.

4.2. Алгоритм створення гітарного тренажера

Можливості покращення розробленого веб-плагіну та його інтеграції з веб-антивірусами або новими системами виявлення та запобігання вторгнень (IDPS) включають в себе ряд стратегій. Зокрема, важливо розширити функціонал плагіну для виявлення нових загроз та покращення його ефективності. Інтеграція з веб-антивірусами може бути поліпшена за допомогою отримання регулярних оновлень та розширення можливостей виявлення шкідливого контенту.

Співпраця з новими IDPS програмами може вимагати стандартизованих інтерфейсів для покращення сумісності та обміну інформацією про загрози. Застосування технологій машинного навчання та штучного інтелекту дозволяє автоматизувати виявлення аномалій та навчати систему адаптуватися до нових видів загроз.

Розглядання можливостей обміну інформацією про загрози (TI) сприятиме швидшому виявленню та реагуванню на нові кіберзагрози. Враховуючи динаміку кібербезпеки, постійна модернізація веб-плагіну та його інтеграція з іншими заходами забезпечать високий рівень безпеки та стійкість до еволюції загроз у веб-мережах.

Для вдосконалення веб-плагіну, його співпраці з веб-антивірусами та новими IDPS програмами, важливо розглянути інтеграцію з системами захисту в реальному часі та розробку механізмів для виявлення та захисту від новітніх загроз.

Розширення інтерфейсів для співпраці з різними рішеннями дозволить забезпечити сумісність та ефективний обмін інформацією між компонентами системи безпеки. Врахування та використання принципів аналізу контексту та машинного навчання дозволить забезпечити точне виявлення загроз та підвищити рівень автоматизації управління кібербезпекою.

Зокрема, важливо створити модуль для аналізу поведінки користувачів, що дозволить виявляти аномалії та надзвичайні події. Оновлення бази даних вірусів та автоматичне виявлення нових загроз сприятиме оперативному реагуванню на кібератаки. Інтеграція з системами обміну інформацією про загрози дозволить отримувати актуальні дані та ефективно взаємодіяти з іншими гравцями в сфері кібербезпеки. Усі ці кроки допоможуть забезпечити комплексний та високоефективний захист веб-мережі, дозволяючи адаптуватися до найновіших викликів кібербезпеки та ефективно протидіяти загрозам.

ВИСНОВКИ

У ході виконання магістерської роботи, було проведено детальний аналіз розробки веб-плагінів та використання IDPS систем для захисту користувачів у мережі. За розробки веб-плагіну для захисту мережі з використанням IDPS сервера було проведено широкий аналіз сучасних методів та інструментів, спрямованих на покращення безпеки в онлайн середовищі. Основна увага була спрямована на інтеграцію IDPS системи для ефективного виявлення та запобігання потенційним загрозам у режимі реального часу.

Обрано оптимальні інструменти для створення веб-плагіну, зокрема використання мов програмування, таких як JavaScript та HTML для фронтенду, а також Python для реалізації бекенду. Це забезпечує зручність в інтеграції та розширенні функціоналу.

Архітектура веб-плагіну передбачає ряд модулів, включаючи модуль взаємодії з IDPS сервером, модуль відстеження дій користувача та модуль аналізу потоків даних. Кожен з цих модулів виконує важливу функцію в системі, сприяючи забезпеченню повноцінного захисту.

Побудовано бекенд IDPS сервера, який використовує мову програмування Python. Його функціонал охоплює аналіз мережевого трафіку, виявлення потенційних атак та прийняття рішень щодо їх запобігання. Використані відомі інструменти для реалізації IDS сервера, такі як Snort або Zeek.

Модуль «трекера дій користувача» розроблений для виявлення аномалій та сигнатур мережевих атак, поповнення бази даних потенційних загроз. Цей модуль дозволяє спостерігати за активністю користувача та вчасно реагувати на незвичайні сценарії.

Запропоновано використання Web API для ефективного обміну даними між веб-плагіном та IDPS сервером. Це забезпечує стандартизований інтерфейс для взаємодії, полегшуючи розширення та модифікацію системи.

Усі ці кроки дозволили створити комплексне рішення для захисту веб-мережі, використовуючи передові технології та інтегруючи елементи машинного навчання та аналізу данних для більш точного та ефективного виявлення кіберзагроз. Створене рішення відкриває широкі можливості для подальших досліджень та вдосконалення в сфері кібербезпеки.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Miguel M. Naive Bayes Classifier in JavaScript [Електронний ресурс] / Mota Miguel. – 2015. – Режим доступу до ресурсу: https://miguelmota.com/blog/naive-bayes-classifier-in-javascript
2. Ridhi T. Predictive Analysis for Risk Reduction in Data Mining [Електронний ресурс] / Thakur Ridhi. – 2019. – Режим доступу до ресурсу: https://www.researchgate.net/publication/334961363_Predictive_Analysis_for_Risk_Reduction_in_Data_Mining
3. Створення додатку Vue [Електронний ресурс] – Режим доступу до ресурсу: https://ua.vuejs.org/guide/essentials/application.html
4. Fotis Adamakis·. 1. VueUse - The library that makes Vue 3 worth the upgrade [Електронний ресурс] / Fotis Adamakis· – Режим доступу до ресурсу: https://medium.com/js-dojo/vueuse-the-library-that-makes-vue-3-worth-the-upgrade-7047c5bb00ce.
5. Cameron Adams. How to develop a Typescript Library [Електронний ресурс] / Cameron Adams – Режим доступу до ресурсу: https://medium.com/@cameronadams1225/how-to-develop-a-typescript-library-ade8d329636.
6. TypeScript: JavaScript With Syntax For Types [Електронний ресурс] – Режим доступу до ресурсу: https://www.typescriptlang.org/.
7. Abiola Farounbi. Getting started with Vue composables [Електронний ресурс] / Abiola Farounbi – Режим доступу до ресурсу: https://blog.logrocket.com/getting-started-vue-composables/.
8. Flanagan, D. (2011). "JavaScript: The Definitive Guide." O'Reilly Media.
9. Mozilla Developer Network. (2021). "Browser extension." [Електронний ресурс] / Mozilla Developer Network – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Mozilla/Add-ons/WebExtensions

10. Rob, W. (2018). "An Introduction to Browser Extensions." [Електронний ресурс] / Rob, W – Режим доступу до ресурсу: https://www.smashingmagazine.com/2018/09/beginners-guide-browser-extensions/.
11. Northcutt, S., Novak, J., & Winters, S. (2013). "Network Intrusion Detection." SANS Institute.
12. Roesch, M. (1999). "Snort - Lightweight Intrusion Detection for Networks." [Електроннийресурс] / Roesch, M. – Режим доступу до ресурсу: https://www.usenix.org/legacy/event/lisa99/full_papers/roesch/roesch.pdf.
13. Schneier, B. (2015). "Data and Goliath: The Hidden Battles to Collect Your Data and Control Your World." W. W. Norton & Company.
14. NIST Cybersecurity Framework. (2018). "Framework for Improving Critical Infrastructure Cybersecurity." [Електронний ресурс] / NIST Cybersecurity Framework. – Режим доступу до ресурсу: https://www.nist.gov/cyberframework.
15. MDN Web Docs. (2021). "Web API." [Електронний ресурс] / MDN Web Docs – Режим доступу до ресурсу: https://developer.mozilla.org/en-US/docs/Web/API.
16. Fielding, R. (2000). "Architectural Styles and the Design of Network-based Software Architectures." [Електронний ресурс] / Fielding, R – Режим доступу до ресурсу: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm
17. Bejtlich, R. (2004). "The Tao of Network Security Monitoring: Beyond Intrusion Detection." Addison-Wesley.
18. Stamp, M. (2006). "Information Security: Principles and Practice." Wiley.
19. Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). "Cybersecurity Threats, Vulnerabilities, and Attacks: Trends and Challenges." IEEE Security & Privacy, 15(1), 38–45.
20. Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). "Deep Learning." MIT Press.

21. Hunt, T., & Gude, R. (2018). "Mastering Web APIs: Expert Techniques for Building and Consuming Web APIs." O'Reilly Media.
22. Stuttard, D., & Pinto, M. (2011). "The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws." Wiley.
23. Brown, Z., & Grimsley, M. (2020). "Building Browser Extensions with Vue.js." [Електронний ресурс] / Brown, Z., & Grimsley, M. – Режим доступу до ресурсу: https://blog.logrocket.com/building-browser-extensions-with-vue-js/