Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Дослідження ризиків та вразливостей в системі керування розумним будинком

Виконав студент 6 курсу, групи СБм-61
спеціальності (напрямку підготовки)
125 «Кібербезпека»
(шифр і назва спеціальності (напряму підготовки))

		Кивацький І.М.
	(підпис)	(прізвище та ініціали)
Керівник		Оробчук О.Р..
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Лечаченко Т.А.
	(підпис)	(прізвище та ініціали))
Завідувач кафедри		Загородна Н.В.
	(підпис)	(прізвище та ініціали)
Рецензент		Жаровський Р.О.
	(підпис)	(прізвище та ініціали)

м. Тернопіль – 2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії

Кафедра кібербезпеки

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Загородна Н.В.
(підпис) (прізвище та ініціали)
« » 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 125 Кібербезпека

студенту Кивацькому Івану Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження ризиків та вразливостей в системі керування розумним будинком

Керівник роботи Оробчук Олександра Романівна доктор філософії,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом по університету від «16» листопада 2023 року № 4/7-1061

2. Термін подання студентом роботи 16.12.2023

3. Вихідні дані до роботи наукові літературні джерела

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1 Аналітична частина. 2 Теоретична частина. 3. Практична реалізація.
4 Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1.Тема, мета, задачі, об'єкт, предмет дослідження. 2. Актуальність. 3. AJAX Systems
4. OWASP Top 10. 5. OWASP IOT TOP 10. 6. SPA .7.MPA. 8.SSR. 9. Memory Leak. 10. ReDoS
11. Мікросервісна архітектура. 12.MongoDB Replica Set 13. Діаграма мікросервісів. 14. NPM
Вразливості. 15. Sentry. 16. JWT. 17. Ризики використання штучного інтелекту.
18. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М. к.т.н доцент		
Безпека в надзвичайних ситуаціях	Клепчик В.М. ст.. викладач		

7. Дата видачі завдання 16.11. 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи	16.11.2023	Виконано
2	Аналіз літературних джерел	16.11.23-20.11.23	Виконано
3	Обґрунтування актуальності дослідження	21.11.23	Виконано
4	Аналіз предмету дослідження та предметної області	23.11.23	Виконано
5	Проведення дослідження методів та засобів аналітичного опрацювання даних	23.11.23 – 26.11.23	Виконано
6	Оформлення розділу «Аналітична частина»	27.11.23-28.11.23	Виконано
7	Оформлення розділу «Теоретична частина»	29.11.23-01.12.23	Виконано
8	Оформлення розділу «Практична частина»	02.12.23-08.12.23	Виконано
9	Оформлення розділу «Охорона праці та безпека в надзвичайних ситуаціях»	09.12.23	Виконано
10	Нормоконтроль	11.12.23	Виконано
11	Попередній захист роботи	20.12.23	Виконано
12	Захист кваліфікаційної роботи	26.12.23	

Студент _____ Кивацький І.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Оробчук О.Р.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Дослідження ризиків та вразливостей в системі керування розумним будинком/ / Кваліфікаційна робота за освітнім рівнем «Магістр»// Кивацький Іван Миколайович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно–інформаційних систем і програмної інженерії, кафедра комп'ютерних систем та мереж, група СБм-61 // Тернопіль, 2023 // с. – 57, рис. – 37, додат. – 4, бібліогр. – 15.

Ключові слова: РОЗУМНИЙ БУДИНОК, ДОСЛІДЖЕННЯ РИЗИКІВ, ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ, SSR MEMORY LEAK, DDOS, AWS CLUSTER, MONGODB REPLICA SET

Кваліфікаційну роботу магістра присвячено дослідженню ризиків та вразливостей в системі керування розумним будинком. Проведено порівняння доступних архітектурних рішень при проектування клієнт-серверної архітектури. Досліджено доступні архітектури та парадигми як для клієнтської частини так і для серверної, порівняно їхні переваги, недоліки, також проаналізовано ризики, вразливості та методи боротьби.

В ході виконання дипломної роботи обрано архітектурні рішення які будуть оптимальними для проектованого додатку, спроектовано мікросервісну архітектуру серверної частини додатку, також спроектовано структуру NoSQL бази даних яка буде використовуватись для зберігання даних її колекції, Проаналізовано можливі вразливості на інфраструктурному рівні, запропоновано рішення потенційних проблем та ризиків.

Окрему увагу приділено ризикам використання штучного інтелекту в додатку, враховано можливі репутаційні та юридичні ризики через неправомірне використання даних користувачів для навчання моделі штучного інтелекту.

ANNOTATION

Research on risks and vulnerabilities in a Smart Home management system. // Master thesis // Kyvatskyi Ivan// Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Cyber Security, СБм-61 group // Ternopil, 2023 // p. – 57, fig. – 37, annexes - 4, Ref. - 15.

Keywords: SMART HOME, RISKS RESEARCH, VULNERABILITIES RESEARCH, SSR MEMORY LEAK, DDOS, AWS CLUSTER, MONGODB REPLICA SET

The qualification work is devoted to the study of risks and vulnerabilities in the smart home management system. A comparison of available architectural solutions for designing client-server architectures is made. The available architectures and paradigms for both the client side and the server side were studied, their advantages and disadvantages were compared, and risks, vulnerabilities and methods of combat were also analyzed.

In the course of the thesis, architectural solutions were chosen that would be optimal for the designed application, the microservice architecture of the server part of the application was designed, the NoSQL database structure was also designed, which will be used to store the data of its collection, possible vulnerabilities at the infrastructure level were analyzed, solutions to potential problems and risks were proposed.

Considerable attention is paid to the risks of using artificial intelligence in the application, taking into account possible reputational and legal risks due to the misuse of user data to train an artificial intelligence model.

ЗМІСТ

ВСТУП 7
I. АНАЛІЗ ДОСЛІДЖЕНЬ У СФЕРІ РИЗИКІВ ТА ВРАЗЛИВОСТЕЙ СИСТЕМ РОЗУМНОГО БУДИНКУ 9
1.1 Важливість систем розумного будинку 9
1.2 Існуючі дослідження 13
II. ПРОЕКТУВАННЯ СИСТЕМИ КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ .. 18
2.1 Архітектура клієнтної частини. 18
2.2 Node.js 25
2.3 MongoDB 28
2.4 Мікросервісна архітектура 34
2.5 Структура мікросервісів 38
III. ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ СИСТЕМИ 40
3.1 Використання компонентів з відомими вразливостями 40
3.2 Логування та відслідковування помилок 43
3.3 Захист від DDOS атак 45
3.4 Ризики використання штучного інтелекту. 47
3.5 Ризики JWT авторизації. 48
IV. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 51
4.1 Охорона праці 51
4.2 Комп’ютерне забезпечення процесу оцінки радіаційної та хімічної обстановки 54
ВИСНОВКИ 56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57
ДОДАТОК А Тези конференції 58

ВСТУП

Актуальність теми. Згідно досліджень у світі сьогодні використовують близько 450 мільйонів розумних будинків, дослідники прогнозують тільки зростання числа у наступні роки. Також важливо що системи розумного будинку застосовують не тільки у приватних помешканнях, але і в багатьох державних та приватних установах, наприклад в лікарнях, де системи допомагають пацієнтам коригувати палату під свої потреби не встаючи з лікарняного ліжка.

Також до систем розумного будинку можна віднести системи охорони та контролю, нескладно уявити наскільки ці системи є важливими і яку важливу функцію виконують.

Кожна з систем розумного будинку потребує системи для керування його елементами, наприклад веб додаток чи мобільний додаток де користувач може змінювати конфігурацію пристроїв, вмикати, вимикати, перевірити їхній став навіть будучи далеко від об’єкту.

Такі системи стикаються з великими ризиками, адже наслідки можуть бути критичні, якщо не смертельні для користувачів. Системи керування розумним будинком часто оперує особистими даними користувачів, їх іменами, мобільними телефонами, адресами, інформацією коли вони виходять і заходять в будинок. Також в залежності від підключених пристроїв може міститись інформації про їхній розпорядок дня, чи навіть хвороби та особливі потреби. Не можна переоцінити ризики які виникнуть у випадку коли зловмисник отримає доступ до системи керування розумним будинком іншого користувача, він зможе відчити, зачинити двері об’єкта, вимкнути камери спостереження, нанести навіть непоправну шкоду.

Зважаючи на ці небезпеки дослідження ризиків та вразливостей таких систем є гостро актуальним, варто зважати що ризики будуть тільки збільшуватись, через розвиток систем, залучення штучних інтелектів до їх керування, та їхню популяризацію.

Мета дипломної роботи є дослідження ризиків та вразливостей системи керування розумним будинком

Для досягнення мети наукового дослідження потрібно виконати наступні завдання:

- опрацювати літературні джерела та переглянути існуючі рішення;
- сформулювати основні проблеми з якими стикалися інші дослідники;
- порівняти можливі архітектури та парадигми, дослідити їх ризики;
- спроектувати систему керування розумним будинком;
- дослідити можливі атаки та вразливості системи;
- запропонувати засоби для зменшення ризиків.

Об’єктом дослідження є процес дослідження, проектування та аналізу ризиків та вразливостей системи керування розумним будинком в клієнт-севрверній архітектурі.

Предметом дослідження є ризики та вразливості в програмі системи керування розумним будинком.

Методи дослідження. Методом дослідження було обрано методу систематичного аналіз, спираючись на те що проблема вразливостей та ризиків в системах розумного будинку з особливими потребами є проблема яка обумовлена великою кількістю факторів які мають досить тісні зв'язки між собою, тому вимушено їх вирішувати методом систематичного аналізу.

Наукова новизна одержаних результатів. Досліджено не тільки комунікаційні та інфраструктурні ризики системи керування розумним будинком, а також програмні, які можуть виникати при неправильно спроектованій архітектурі додатку.

Практична цінність результатів дослідження. Дослідження ризиків та вразливостей дозволить покращити захищеність систем керування розумним будинком, та ускладнить реалізацію атак на такі системи.

І. АНАЛІЗ ДОСЛІДЖЕНЬ У СФЕРІ РИЗИКІВ ТА ВРАЗЛИВОСТЕЙ СИСТЕМ РОЗУМНОГО БУДИНКУ

1.1 Важливість систем розумного будинку

Розумний будинок – це зручна домашня система, де побутові прилади та пристрої можуть бути автоматично керовані дистанційно з будь-якого місця з доступом до Інтернету за допомогою персонального комп'ютера, мобільного телефону чи іншого мережевого пристрою. Пристрої в розумному будинку взаємодіють через Інтернет, що дозволяє користувачеві дистанційно керувати функціями, такими як безпека доступу до будинку, температура, освітлення та камери відеоспостереження (рис.1.1).

Рисунок 1.1 – Система розумний будинок

Використання розумних будинків зростає щорічно, до 2028 року прогнозують понад 780 мільйонів таких систем, на 2023 таких систем близько 450 мільйонів. Розумні пристрої впевнено витискають своїх попередників з ринку(рис 1.2), наприклад зараз уже рідко купують звичайний світильник, тепер майже кожен з них має пульт для керування, підключення по Wifi/ Bluetooth.

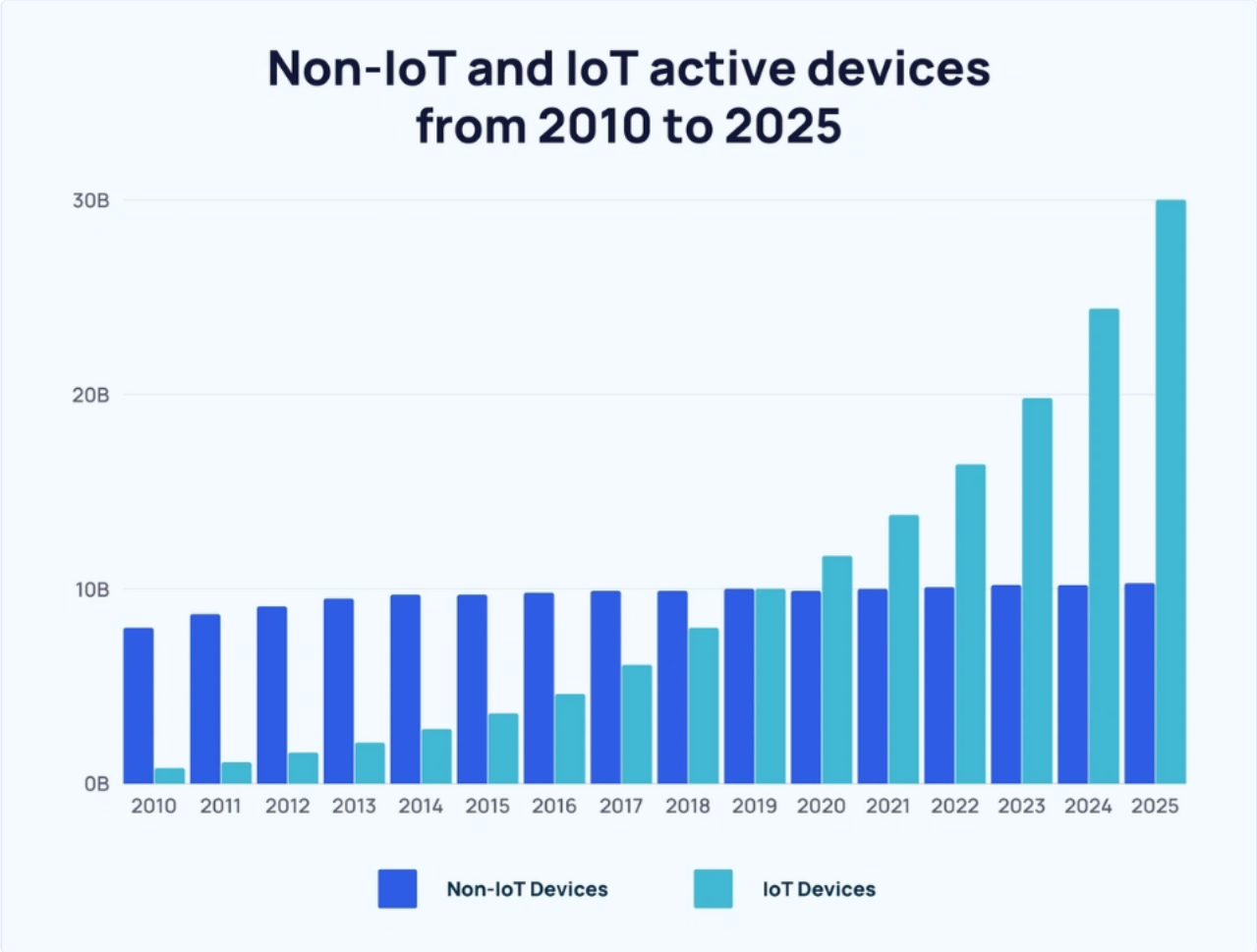

Рисунок 1.2 – Частка IoT пристроїв зростає з кожним роком

Пристрої розумного будинку бувають обладнані навичками самонавчання(штучний інтелект), щоб вони могли вивчати розклади власника будинку і вносити зміни за необхідності. Розумні будинки із системами керування освітленням дозволяють власникам зменшити споживання електроенергії та скористатися економією витрат на енергію.

Деякі системи автоматизації будинку повідомляють власника, якщо вдома виявлено будь-який рух у його відсутність, тоді як інші можуть викликати правоохоронні органи, поліцію чи пожежну службу у випадку надзвичайних ситуацій.

Після підключення послуги, такі як розумний дзвінок, розумна система безпеки та розумні прилади, є частиною технології Інтернету речей (IoT) – мережі фізичних об'єктів, які можуть збирати та обмінювати електронною інформацією.

Однією з найбільш обґрунтованих характеристик розумного будинку є покращені можливості безпеки. Багато продуктів тепер обладнані камерами, які відстежують рух, записують відео або дозволяють вести пряму відео трансляцію. Це може бути встановлено для синхронізації з дзвінком дверного дзвінка або налаштовано для відображення на певних ділянках вашої власності. Ці відеозаписи можуть дозволяти відео дзвінки з особою біля вашого входу, включаючи аудіо функції.

Багато розумних будинків також обладнані сучасними системами безпеки. Це включає в себе детектори руху, які виявляють присутність осіб у випадках, коли їх там не повинно бути, системи моніторингу дому, сповіщення і сигнали про підозрілу поведінку, а також можливість віддалено закривати двері або вікна за допомогою телефону.

Успішним представником систем розумних будинків є український продукт AJAX System, це компанія, яка спеціалізується на створенні систем безпеки та охорони. AJAX Systems розробляє та виробляє інноваційні системи безпеки для захисту будинків, офісів та інших об'єктів[11].

Рисунок 1.3 – Система AJAX Systems

До переваг AJAX Systems можна віднести:

- бездротова технологія, AJAX використовує передову бездротову технологію, що дозволяє швидку та зручну інсталяцію системи без потреби провідного підключення;
- надійність і стабільність, системи AJAX відомі своєю надійністю та стабільністю роботи. Вони використовують захищені канали зв'язку, що робить їх стійкими до перешкод та втручань;
- широкий функціонал, сервіс охоплює різні аспекти безпеки, включаючи виявлення руху, дверей/вікон, контроль доступу, виявлення витоків газу та води, димові та газові детектори, а також можливості відеоспостереження;
- мобільний додаток, користувачі можуть контролювати свою систему безпеки через мобільний додаток, надаючи зручний та віддалений доступ;
- інтеграція з іншими системами, система може інтегруватися з іншими сучасними технологіями та системами автоматизації, що розширює їхні можливості;
- інтеграція з іншими системами, сервіс може інтегруватися з іншими сучасними технологіями та системами автоматизації, що розширює їхні можливості.

1.2 Існуючі дослідження

Дослідженням ризиків та вразливостей в веб додатках досліджував проект і продовжує досліджувати консорціум OWASP котрий розробив рекомендації OWASP TOP 10.

Open Web Application Security Project (OWASP) є некомерційною організацією, яка спрямована на підвищення безпеки програмного забезпечення.

OWASP зосереджується на розробці рекомендацій, стандартів та інструментів для забезпечення безпеки веб-додатків.

Основні цілі OWASP включають:

- усвідомлення ризиків, збільшення усвідомлення розробників, тестувальників та менеджерів про ризики безпеки веб-додатків;
- стандарти та інструменти, розробка стандартів безпеки та надання інструментів, які допомагають у виявленні та виправленні уразливостей у веб-додатках;
- навчання та ресурс, надання навчальних матеріалів, документації та ресурсів для підвищення рівня знань у галузі безпеки програмного забезпечення;
- заохочення найкращих практик, сприяння впровадженню найкращих практик в галузі безпеки веб-додатків та розробки програмного забезпечення.

Організація регулярно публікує OWASP Top Ten(рис 1.4), список найбільш критичних уразливостей у веб-додатках, що допомагає фахівцям з безпеки та розробникам зосередити увагу на найважливіших аспектах безпеки. Крім того, OWASP розробляє різні проекти та інструменти, такі як OWASP ZAP (Zed Attack Proxy) та інші, для виявлення та усунення уразливостей веб-додатків[10].

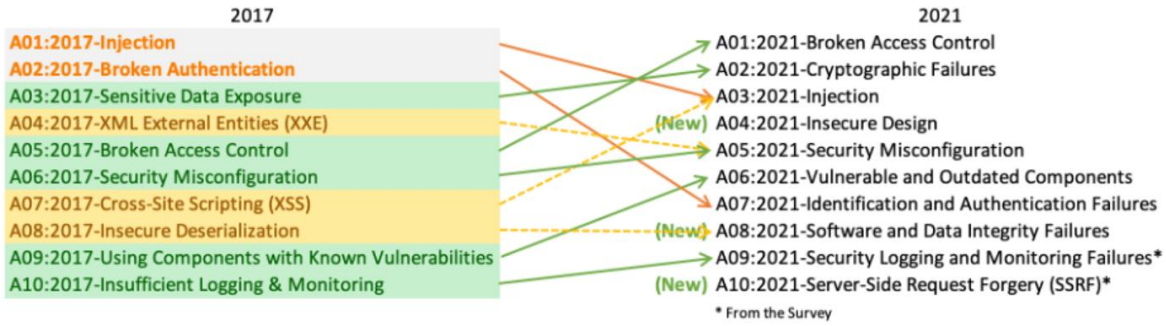

Рисунок 1.4 – OWASP Top 10 2021

OWASP Top 10 - це список найбільш критичних уразливостей у веб-додатках, який регулярно публікується OWASP. Цей список служить орієнтиром для розробників, тестувальників та фахівців з безпеки для виявлення та усунення загроз у сфері веб-безпеки. Нижче коротко розглянуто пункти OWASP Top 10 2021:

- injection: уразливість, яка дозволяє атакуючому впроваджувати в шлях або дані введення зловмисний код, що може використовуватися для отримання неправомірного доступу до бази даних;
- broken authentication: некоректна реалізація механізмів аутентифікації, що може призвести до неправильного управління сесіями, паролями та іншими елементами безпеки;
- sensitive data exposure: недостатній захист чутливих даних, таких як паролі чи особиста інформація, що може призвести до їхнього витоку;
- xml external entities, атаки, пов'язані з обробкою зовнішніх сутностей XML, що може використовуватися для витоку чутливої інформації чи виклику інших атак;
- broken access control: вразлива реалізація механізмів контролю доступу, що дозволяє атакуючим отримати доступ до неавторизованих ресурсів;
- security misconfiguration: неправильна конфігурація серверів, платформ та додатків, що може призвести до витоку чутливої інформації;

- cross-site scripting: уразливість, яка дозволяє вбудовувати скрипти на сторінках, що відображаються іншим користувачам, приводячи до виконання зловмисного коду в їхньому браузері;
- insecure deserialization: несправна обробка серіалізованих об'єктів, що може призвести до виконання зловмисного коду під час їхнього відновлення;
- using components with known vulnerabilities: використання сторонніх бібліотек чи компонентів із відомими уразливостями, що може призвести до атак;
- insufficient logging and monitoring: неправильне ведення журналів та недостатній моніторинг подій, що ускладнює виявлення та реагування на потенційні атаки.

Цей список регулярно оновлюється, відображаючи найактуальніші загрози веб-безпеці.

OWASP IoT (Open Web Application Security Project Internet of Things) – це ініціатива OWASP, яка фокусується на забезпеченні безпеки Інтернету речей (IoT). OWASP IoT працює над ідентифікацією та розробкою рекомендацій для захисту IoT-продуктів від потенційних загроз і атак[10].

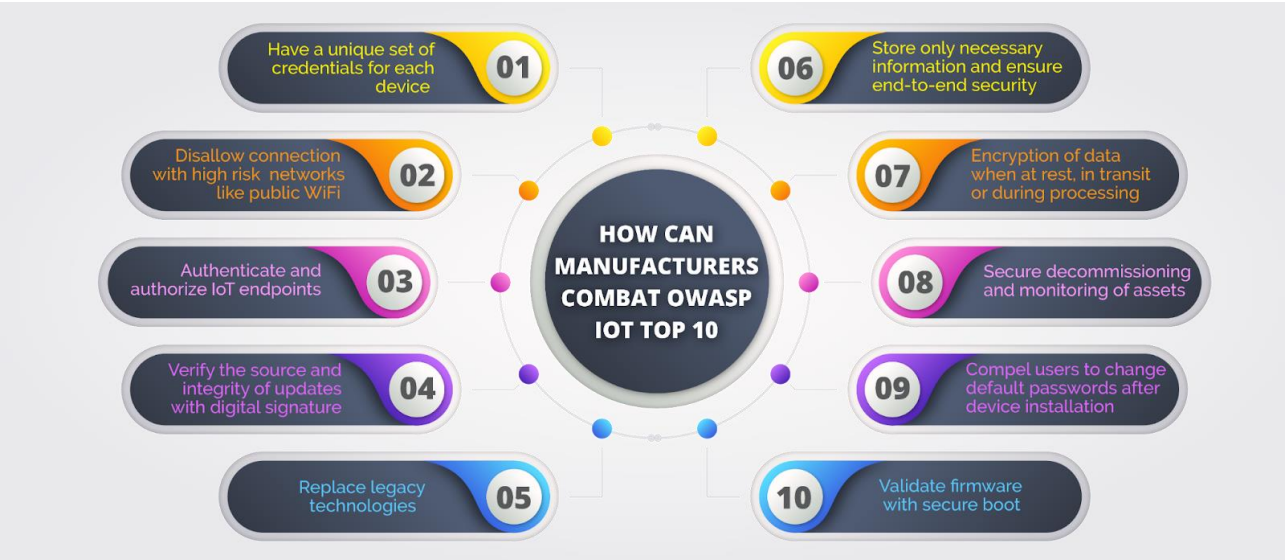

Рисунок 1.5 – Рекомендації для OWASP IOT Top 10

OWASP IoT Top Ten є списком найбільш критичних уразливостей в Інтернеті речей (IoT). Кожен елемент цього списку підкреслює конкретний ризик або проблему в безпеці IoT.

Нижче коротко розглянуто основні пункти OWASP IOT Top 10

OWASP IoT Top Ten:

- weak, guessable, or hardcoded passwords уразливості, пов'язані з використанням слабких, легко вгадуваних паролів, що може призвести до неправомірного доступу;
- insecure network services (ненадійні мережеві сервіси): проблеми безпеки, пов'язані з ненадійними або невірно налаштованими мережевими сервісами IoT-пристроїв, які можуть бути використані для атаки;
- insecure ecosystem interfaces (ненадійні інтерфейси екосистеми): пов'язані з ненадійними інтерфейсами та API, які забезпечують взаємодію між IoT-пристроями та іншими компонентами;
- lack of secure update mechanism (відсутність безпечного механізму оновлення): проблеми, які виникають внаслідок відсутності безпечного механізму оновлення програмного забезпечення на IoT-пристроях.
- use of insecure or outdated components (використання ненадійних або застарілих компонентів): ризики, пов'язані з використанням ненадійних або застарілих компонентів у програмному забезпеченні IoT-пристроїв;
- insufficient privacy protection (недостатній захист приватності):проблеми, які виникають внаслідок недостатнього захисту особистих даних користувачів, які збираються та обробляються IoT-пристроями;
- insecure data transfer and storage (ненадійне зберігання даних): ризики, пов'язані з ненадійним зберіганням даних, які обмінюються між IoT-пристроями та іншими системами;

- lack of device management (відсутність управління пристроями): проблеми, які виникають через відсутність ефективного управління IoT-пристроями, включаючи недостатню можливість відключення та ізоляції;
- insecure default settings (ненадійні стандартні налаштування): вразливості, пов'язані з ненадійними стандартними налаштуваннями, які можуть залишатися без змін після встановлення;
- lack of physical hardening (відсутність фізичного зміцнення): проблеми, які виникають через недостатній фізичний захист IoT-пристроїв від фізичного доступу.

На мою думку в дослідженнях OWASP не достатньо розглянуто програмні ризиків та вразливості які можуть виникнути при використанні неправильної архітектури однієї з частин клієнт серверної архітектури, та недостатній кваліфікації розробників, гарним прикладом є SSR Memory Leak(пункт 2.1).

Проаналізовано проблему дослідження ризиків та вразливостей систем керування розумним будинком у результаті якого виявлено, що дослідження ризиків та вразливостей є актуальною задачею і потребує додаткового дослідження.

Проведено аналіз сучасних підходів та існуючих досліджень щодо покращення ризиків та вразливостей систем керування розумним будинком середовищ. На основі аналізу існуючих рішень встановлено, що існуючі дослідження оминають певні аспекти програмних ризиків.

II. ПРОЕКТУВАННЯ СИСТЕМИ КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ

2.1 Архітектура клієнтної частини.

Для клієнтської частини у браузері вирішено використовувати Single Page Application(SPA). Single Page Application (SPA) мають кілька переваг, які роблять їх популярнішими в порівнянні з Multiple Page Application(MPA) чи Server Side Render(SSR)(рис.2.1).

- Швидкість та ефективність:SPA завантажують усі необхідні ресурси при початковому вході на сайт. Після цього вони працюють з усіма запитам за допомогою AJAX (Asynchronous JavaScript and XML), уникаючи при цьому без перезавантаження сторінки, яке створює певні незручності при користуванні продуктом. Також відсутність повторного завантаження сторінок дозволяє зменшити затримку і покращити користувацький досвід[1].
- Зменшення серверних навантажень:Споживання трафіку зменшується у рази завдяки тому, що сервер повертає тільки статичні файли які потрібні конкретній сторінці сайту, для рендерингу контенту, Також запити до сервера здійснюються асинхронно, що при правильному підході до написання серверної частини дозволяє покращити продуктивність.

Використання популярних фреймворків і бібліотек для SPA (наприклад, React, Angular, Vue.js) забезпечує консистентність та швидкість розробки[8].

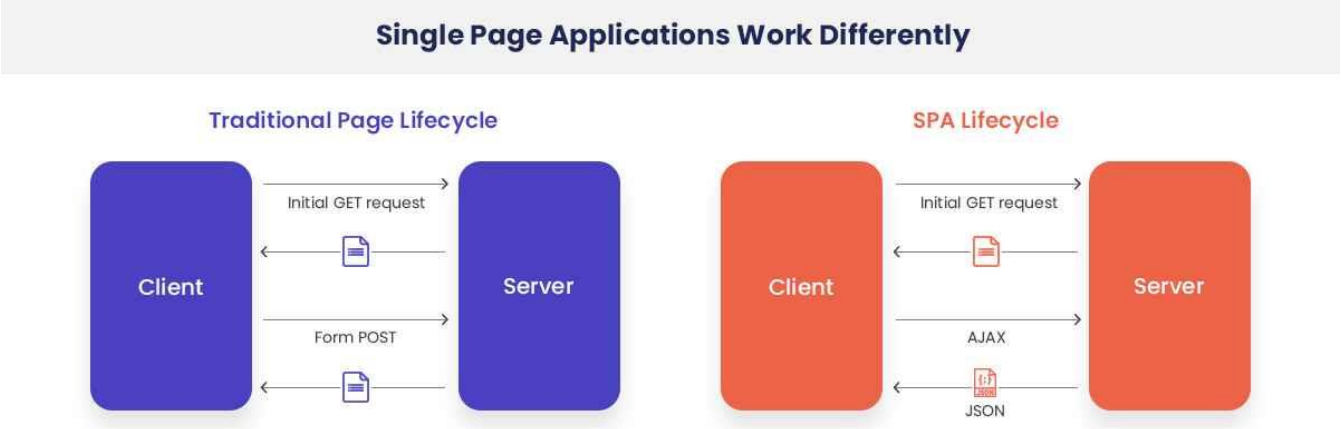

Рисунок 2.1 – Порівняння MPA та SPA

SPA також мають свої значні недоліки, одним з ключових є проблеми з Search Engine Optimization(SEO). Це є відомою проблемою сучасних SPA, оскільки контент сторінки створюється динамічно, Googlebot не можу проаналізувати HTML веб-сайту оскільки він створюється динамічно на клієнтській частині. Якщо для вашого продукту важливо SEO, варто звернути увагу на SSR, який має гібридну архітектуру, на серверній частині генерується початкове HTML документ для сторінки, з можливістю виконання запитів на стороні сервера для отримання даних для побудови документу, і уже на клієнтській частині підключається фреймворк для роботи з Virtual DOM(рис. 2.2)

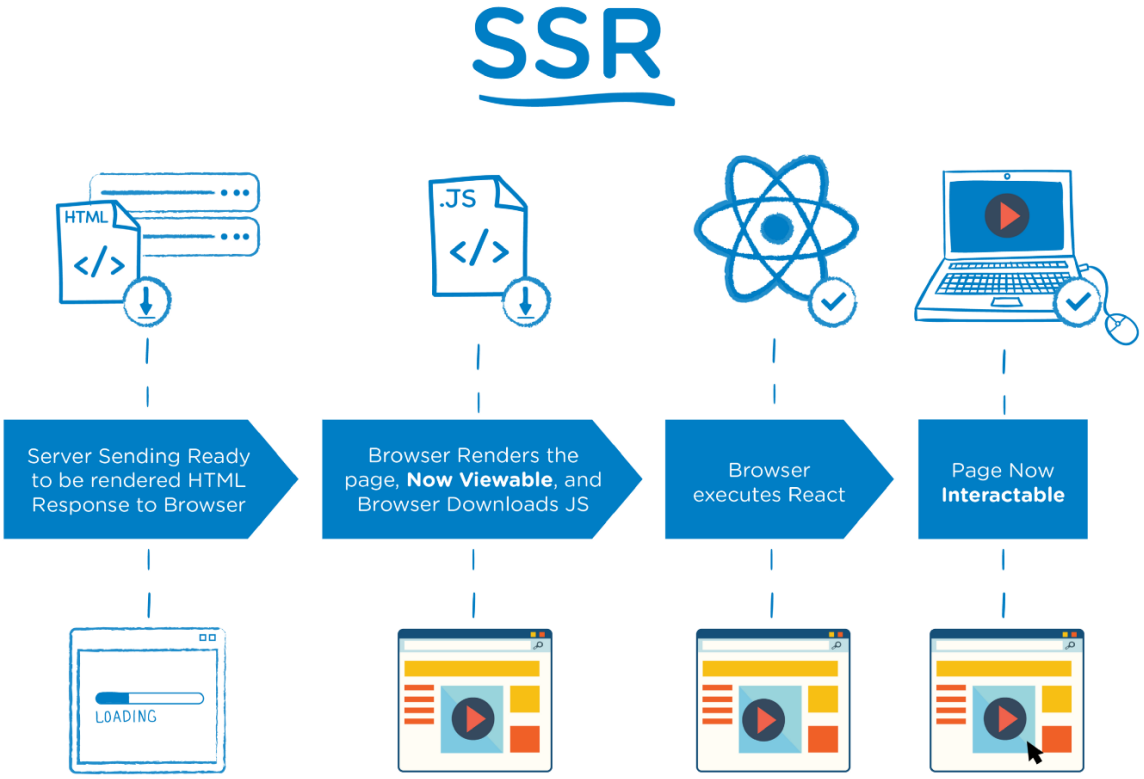

Рисунок 2.2 – Server Side Render

Використання SSR створює певні ризики, наприклад значно складніший deploy додатку.

У випадку використання Server Side Render збільшується в рази навантаження на сервер, це означає що сервер повинен використовувати значно потужніший процесор і об’єм оперативної пам'яті. Також варто подумати про використання кластерів з авто розгортанням, який буде реагувати на навантаження та запускати додаткові інстанси для обробки великої кількості запитів[12].

Також збільшуються ризики через помилки у програмній реалізації.

У Javascript робота з пам'яттю відбуваються автоматично, не потрібно виділяти память при створенні змінних і не потрібно очищати пам’ять вручну, коли змінні уже не потрібні. Для очищення пам’яті Javascript використовує Garbage Collector. Він використовує алгоритм “Mark-and-sweep algorithm.

Цей алгоритм зводить визначення «об’єкт більше не потрібний» до «об’єкт недоступний».

Цей алгоритм передбачає дослідження набору об’єктів, які називаються коренями(root). У Javascript корінь є глобальним об’єктом. Періодично збирач сміття буде починати з цих коренів, знаходити всі об’єкти, на які є посилання з цих коренів, потім усі об’єкти, на які посилаються з них, і т.д[8].

Починаючи з коренів, Garbage Collector, таким чином, знаходить всі доступні об’єкти та очищатиме усі недоступні об'єкти.(рис.2.3)

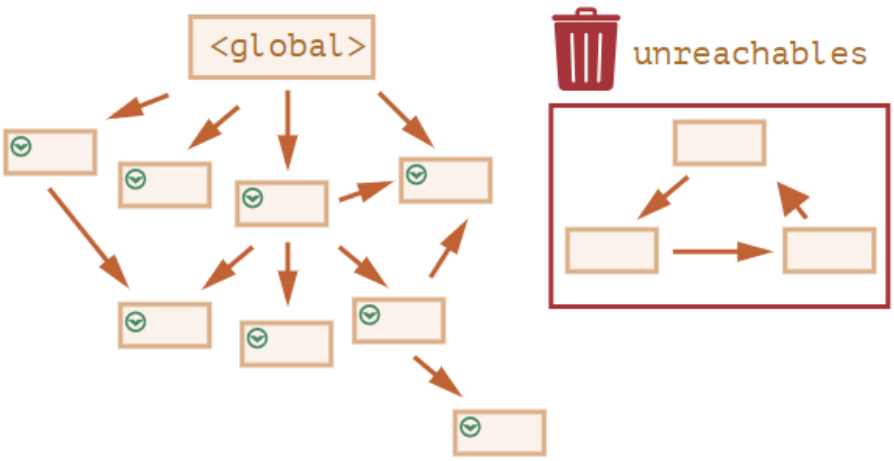

Рисунок 2.3 – Garbage Collector

Однак через те що пам’ять очищається за відомим алгоритмом його можна деколи обійти для реалізації патерну Closure(Замикання), що створює обгортки функції для уникнення очищення пам’яті.(рис.2.4)

Також варто згадати що Javascript використовує модель подій, існують події та їхні слухачі. Важливо очищати слухачі подій коли вони уже не актуальні, оскільки вони також залишаються в пам’яті[2].

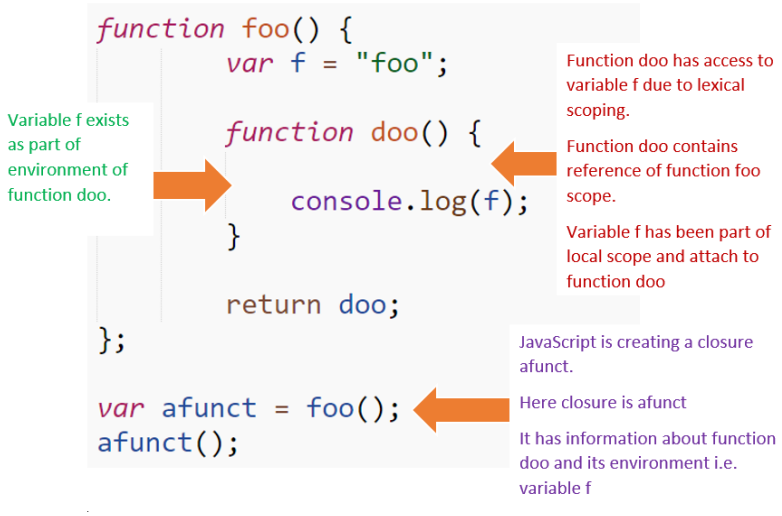

Рисунок 2.4 – Паттерн Closure.

Використання Closure та не очищення слухачів подій може призвести до Memory Leak, це процес коли пам’ять не очищається і залишається в пам’яті, у випадку Single Page Application невеликий Memory Leak буде мати в рази менші наслідки, оскільки він буде працювати тільки в одного клієнта то після закриття вікна з сайтом пам’ять очиститься і користувач при наступному вході на сайт не буде мати зайвої пам'яті з минулого сеансу[2].

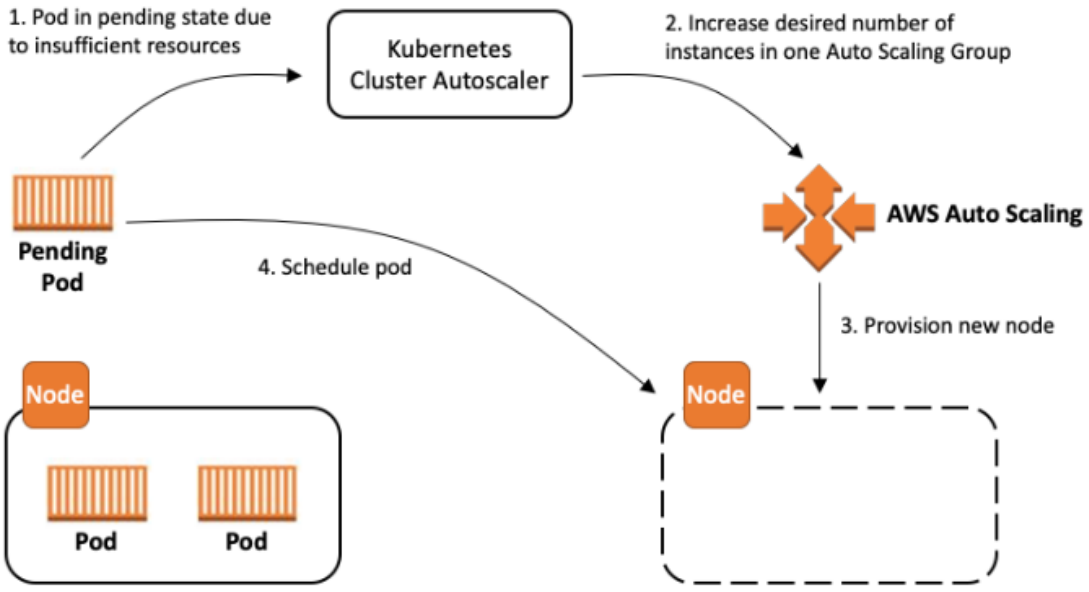

Рисунок 2.5 – AWS cluster Autoscaler

В свою чергу у випадку мінімального Memory Leak на серверній стороні в Server Side Render підході ця пам’ять буде накопичуватись, і призведе до того що сервер не зможе працювати, і стане недоступним. У випадку якщо такий Memory Leak існує, потрібно час щоб його локалізувати і вирішити, щоб додаток міг працювати навіть з таким Memory Leak потрібно використовувати кластер тоді якщо один інстанс перестає відповідати чи досягає критичного рівня оперативної пам'яті, він буде замінений новим інстансом.

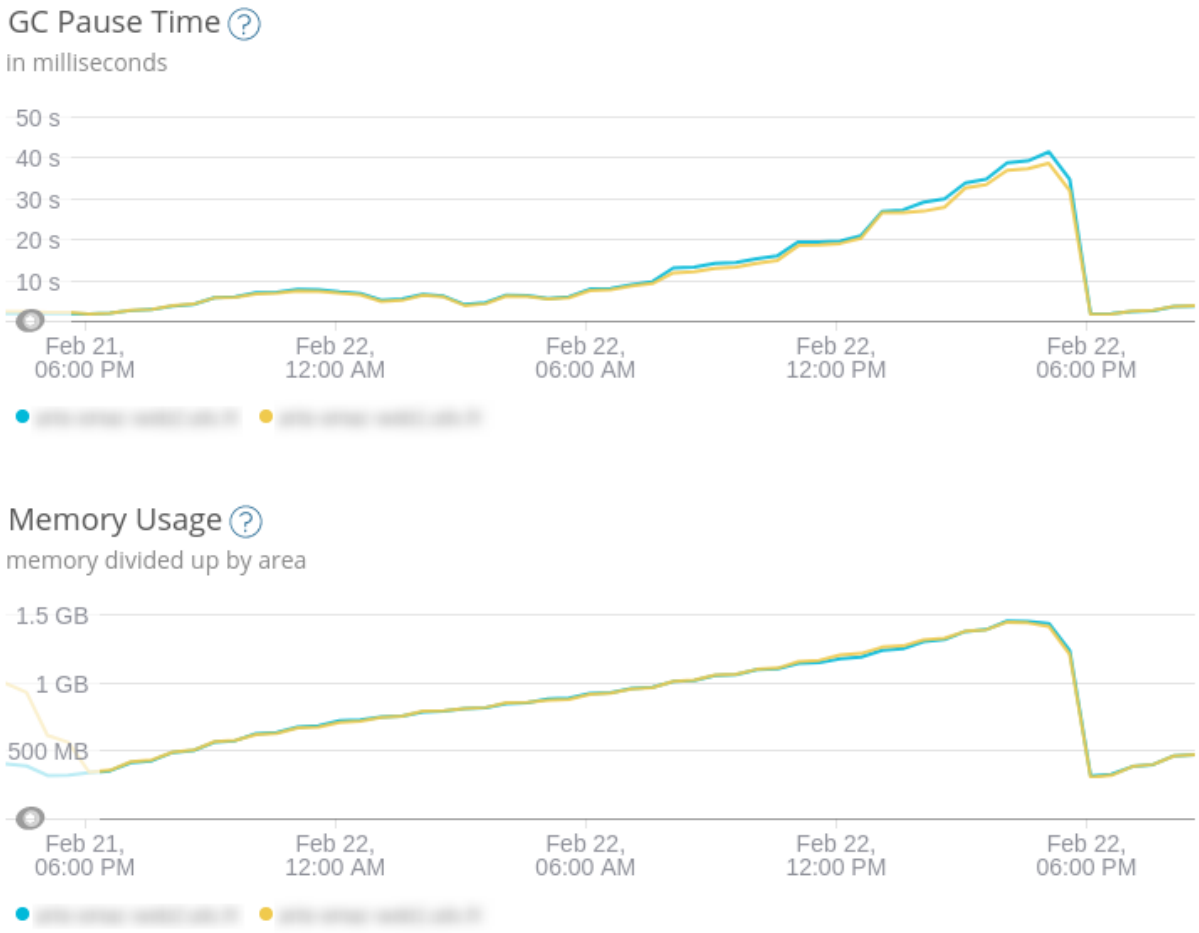

Рисунок 2.6 – Приклад показників які свідчать про Memory Leak

Це дозволить підтримувати додаток у робочому стані поки проблема з Memory Leak вирішується.

Також варто згадати про Regular expression Denial of service (ReDos) атаку, це є типом атак на регулярні вирази, які можуть використовуватися для

зловмисників для завдання великого навантаження на сервер. Ці атаки можуть виникати як у Single Page Applications (SPA), так і у Server-Side Rendering (SSR) додатках, але давайте розглянемо їхні особливості у кожному випадку(рис.2.7).

ReDos атаки у SPA регулярні вирази можуть використовуватися на клієнтській стороні для обробки та перевірки введення користувача або для роботи з даними, отриманими від сервера. Атаки з можуть виникнути, якщо регулярний вираз створений не оптимально і він має експоненціальне зростання, та дозволяє велику кількість можливих шляхів співпадінь[10].

Зловмисники можуть використовувати це для введення даних, які спричиняють велику кількість можливих шляхів співпадінь, і при цьому роблять обробку цих даних витратною від точки зору обчислювальних ресурсів. Це може призвести до блокування або значного сповільнення виконання коду на клієнтському боці.

В Server-Side Rendering, де генерація вмісту відбувається на сервері перед відправленням його клієнту, атаки ReDos можуть виникати при обробці запитів на сервері, які включають в себе регулярні вирази, або наприклад валідації url під час роутингу.

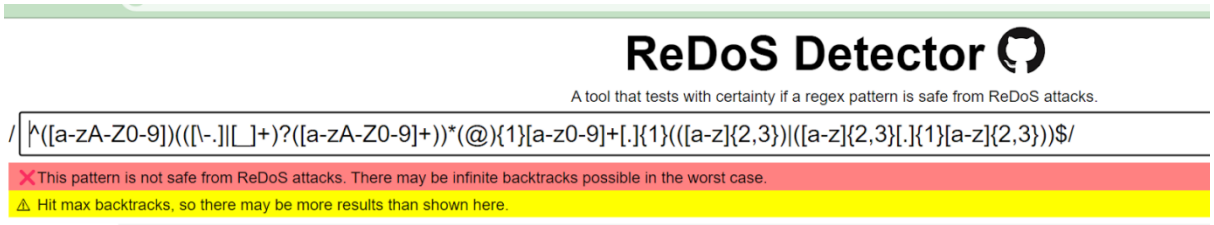

Рисунок 2.7 – Приклад ReDos атаки

Такі атаки можуть призвести до витрат обчислювальних ресурсів на сервері, а в результаті - до відмови в обслуговуванні (DoS) або значного зниження продуктивності сервера.

2.2 Node.js.

Node.js - це середовище виконання для JavaScript, яке дозволяє виконувати код на стороні сервера. Воно використовує двигун V8, який розробляється Google для використання у веб-браузері Chrome. Node.js дозволяє вам використовувати JavaScript не тільки у браузері, але і за його межами, що розширює його використання на серверах та інших областях[3].

Node.js є одно потоковим і використовує асинхронні виклики, що дозволяє обробляти велику кількість запитів без блокування інших операцій. Це робить його ефективним для розробки масштабованих та ефективних за ресурсами додатків.

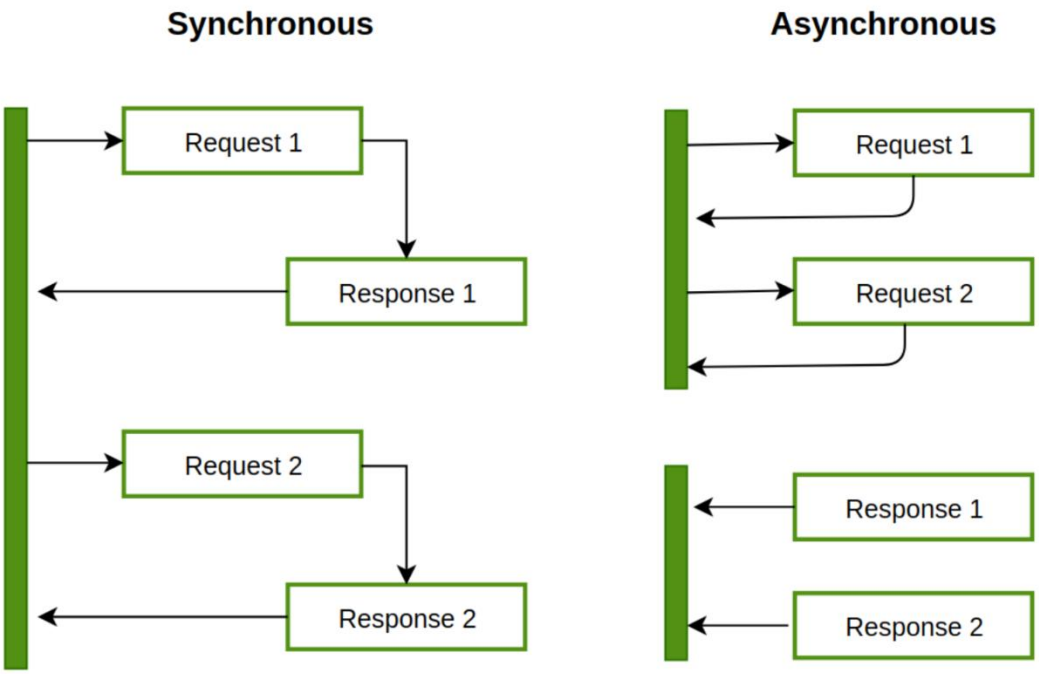

Рисунок 2.8 – Порівняння синхронної та асинхронної моделі

Якщо порівнювати одно потоковий асинхронний підхід та багато потоковий синхронний у кожного з них є свої ризики (рис.2.8).

До ризиків одно потокової асинхронної моделі можна віднести:

- Callback Hell:Велика вкладеність зворотних викликів може призвести до значного ускладнення сприйняття, розуміння та підтримки коду, що називається Callback Hell або Pyramid of Doom;

```
// Callback Hell


a(function (resultsFromA) {
    b(resultsFromA, function (resultsFromB) {
        c(resultsFromB, function (resultsFromC) {
            d(resultsFromC, function (resultsFromD) {
                e(resultsFromD, function (resultsFromE) {
                    f(resultsFromE, function (resultsFromF) {
                        console.log(resultsFromF);
                    })
                })
            })
        })
    })
});

```

у

Рисунок 2.9 – Callback Hell

Проте у Ecmascript 6 у Javascript додано новий інтерфейс Promise, який дозволяє уникати Callback Hell(рис.2.10)[13].

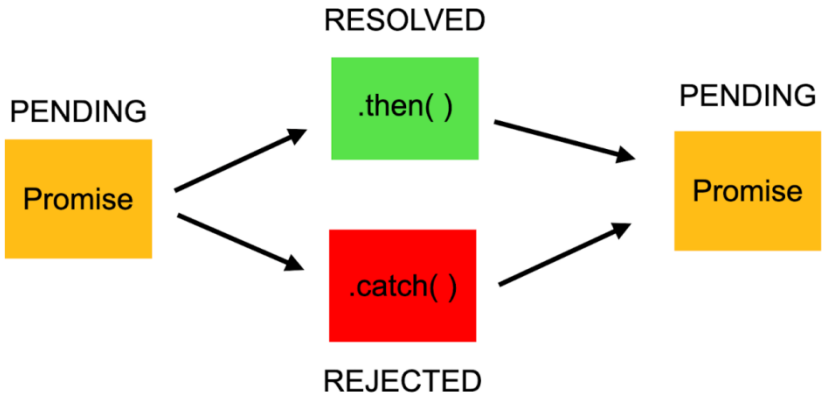

Рисунок 2.10 – Інтерфейс Promise

• система обробки помилок: управління помилками у великих асинхронних програмах може бути складним, оскільки ми використовуємо один потік, при необробленій помилці додаток може зупинити роботу. Уникненням ризику є використання обробки помилок через зворотні виклики try...catch для promise та async await[2].

• відсутність паралельності для обчислень: в інтенсивних обчислюваннях, де важлива паралельність, Node.js може бути менш ефективним. Прикладом є реалізація алгоритмів шифрування та дешифрування. Для вирішення інтенсивних обчисленнях потрібно використовувати відокремлені процеси.

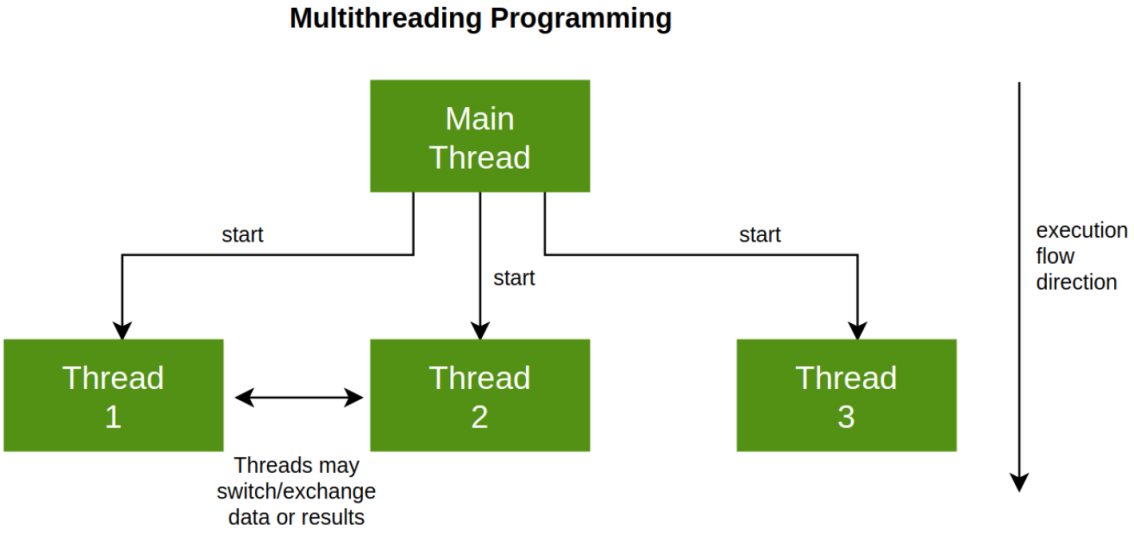

Рисунок 2.11 – Багатопотокова модель роботи

У синхронного багато потокового(рис.2.11) рішення є теж свої ризики:

• блокування потоків: у випадку коли один потік блокується через виконання довгої операцією, він перестає відповідати і це може призвести до блокування не тільки цього потому але і інших потоків та і у можливості затримок у відповіді на інші запити[1];

- управління пам'яттю: у випадку Node.js немає потреби в ручному виділення та очищенні пам’яті, оскільки цим займається Garbage Collector, в свою чергу коли ми використовуємо рішення багатьма потоками виникає ризик конфліктів пам'яті та інших проблем синхронізації. Для їх уникнення потрібно використовувати спеціальні механізми для синхронізації між потоками;[2]
- складність коду: писати безпечний, відмово стійкий та правильно синхронізований код для багато поточного середовища може бути складним. Щоб спростити написання програми з в використанням багато поточності слід використовувати атомарні операції та конструкції для синхронізації уникнення проблем конкурентності.

2.3 MongoDB

MongoDB є базою даних яка використовує NoSQL підхід до побудови баз даних, в цьому підході відсутні схеми, таблиці, запити SQL, ключі та інші компоненти які властиві SQL базам даних(рис.2.12).

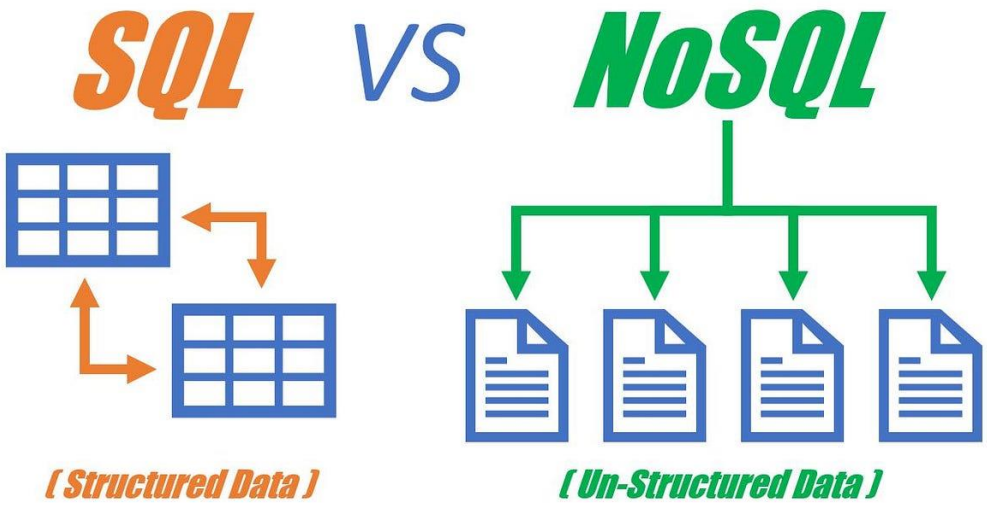

Рисунок 2.12 – Порівняння структури SQL і NoSQL

На відміну від реляційних баз даних MongoDB використовує документно-орієнтовану модель даних, що може дати відчутний приріст у масштабованості та швидкості роботи. MongoDB оперує JSON-подібними документів, котрі зберігаються у форматі BSON (Binary JSON)[4].

Дані організовані у колекції, які є аналогіями таблиць у реляційних базах даних. Колекції зберігають документи(рис.2.13), документ в колекції містить набір ключ-значення, де значення може бути простим типом, масивом чи вкладеним документом. Важливо розуміти що документи можуть, відрізнятись один від одного структурою. Деякі поля можуть бути обов'язковими, деякі - ні.

Рисунок 2.13 – Приклад документу у MongoDB

Операції вставки, оновлення та видалення є атомарними на рівні документів. MongoDB гарантує атомарні операції для одного документа, але не для декількох документів.

MongoDB підтримує індексацію для покращення швидкодії запитів. Індекси можуть бути створені для будь-якого поля в документі.

Важливою перевагою MongoDB яка знижує ризик втрати даних є підтримування різних механізмів збереження, включаючи дискові файли, WiredTiger та MMAPv1. WiredTiger за замовчуванням у нових версіях MongoDB і пропонує кращу продуктивність та керування пам'яттю[4].

Використовуючи бази даних потрібно пам’ятати про ризик втрати даних, щоб зменшити його імовірність MongoDB пропонує ReplicaSet (рис.2.14).

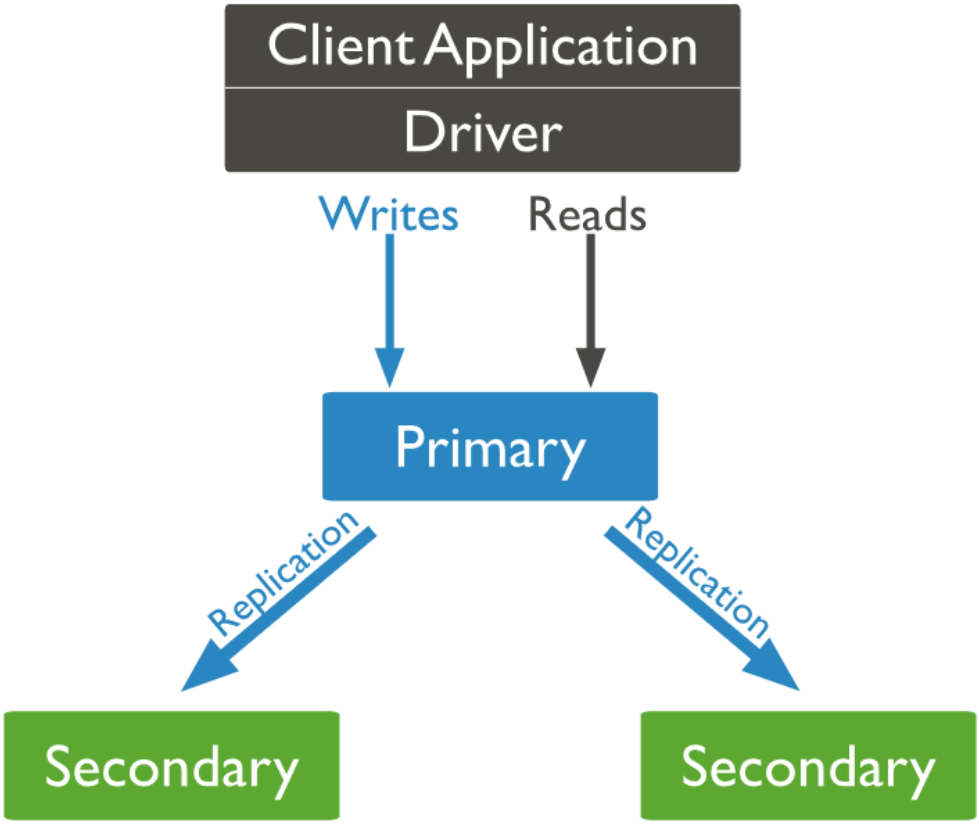

Рисунок 2.14 – MongoDB ReplicaSet

Реплікація в MongoDB є механізмом, який дозволяє підтримувати кілька копій даних та забезпечує високу доступність та стійкість системи. Основною ідеєю реплікації є створення копій основних даних на одному чи кількох другорядних вузлах (secondary nodes), що дозволяє забезпечити продовження роботи системи в разі відмови основного вузла.

Основні етапи роботи реплікації MongoDB:

- ініціалізація реплікації: реплікація починається з того, що один з вузлів обирається як основний (primary). Після цього один або декілька інших вузлів налаштовуються як другорядні (secondary);
- передача даних: першим етапом реплікації є передача даних з основного вузла на другорядні. Це може відбуватися як асинхронно, тобто в основний вузол записуються дані, і пізніше вони реплікуються на другорядні;

- журнал операцій (Oplog): операції, які виконуються на основному вузлі, реєструються в спеціальному журналі операцій, відомому як oplog. Цей журнал служить для того, щоб другорядні вузли могли відтворити ті самі операції у тому ж порядку;
- асинхронна реплікація: другорядні вузли зчитують та відтворюють операції з oplog асинхронно. Це означає, що є певне затримка між тим, як дані записуються на основному вузлі та тим, як вони стають доступними на другорядних;
- читання з другорядних вузлів: коли другорядний вузол відтворює всі операції і доганяє основний вузол, його можна використовувати для читання даних. Це полегшує розподілення навантаження на систему;
- вибір нового основного вузла: У випадку відмови основного вузла система автоматично обирає новий основний вузол з числа налаштованих другорядних;

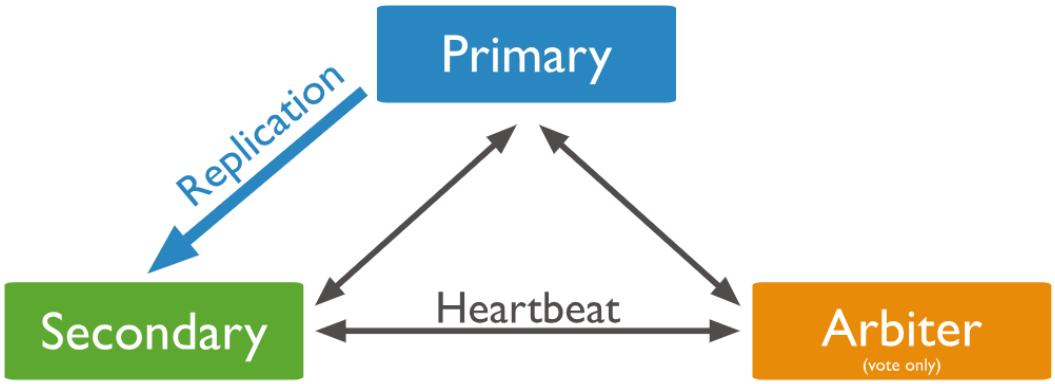

Рисунок 2.15 – Приклад вибору нового основного вузла

- інтеграція з аутентифікацією та авторизацією: реплікація повинна бути налаштована для використання механізмів аутентифікації та авторизації для забезпечення безпеки обміну даними між вузлами;

Мікросервіси використовують чотири колекції MongoDB:

- колекція користувачів використовується для зберігання даних та виконання авторизації користувачів;

```
    firstName: { type: String, trim: true },
    secondName: { type: String, trim: true },
    geoPosition: {
        type: {
            default: 'Point'
        },
        coordinates: {
            lat: { type: Number },
            lng: { type: Number }
        }
    },
    phone: { type: Number },
    password: { type: String, required: true },
    email: { type: String, required:  true, unique: true },
    isAdmin: { type: Boolean, default: false },
    isActivated: { type: Boolean, default: false }
}, { timestamps: { createdAt: 'created_at' } });
```

Рисунок 2.16 – Схема колекції користувачів Це лістинг чи рисунок?

- колекція замовлень зберігає дані про замовлення користувачів, та їхній статус;

```
    user: { type: String, required: true },
    issue: { type: String },
    comment: { type: String },
    typeOfReport: { type: String, required: true },
    geoPosition: {
        type: {
            default: 'Point'
        },
        coordinates: {
            lat: { type: Number },
            lng: { type: Number }
        }
    },
    type: { type: String, required: true },
    status: { type: String, default: 'new' },
    idOfDevice: {
        type: String, required: true
    }
}, { timestamps: { createdAt: 'created_at' } })
```

Рисунок 2.17 – Схема колекції замовлень

- схема налаштувань зберігає дані про налаштування всіх пристроїв;

```
user: { type: String, required: true, unique: true },
elements: [
    {
        type: { type: String, required: true },
        name: { type: String, required: true },
        settings: {
            start: { type: Date },
            end: { type: Date },
            count: { type: Number },
            duration: { type: Number },
            weatherSupport: { type: Boolean, default: false }
        },
        isActive: { type: Boolean, default: true },
        autoSetup: { type: Boolean, default: false },
        motherBoardId: { type: String, default: '' }
    }
],
```

Рисунок 2.18 – Схема колекції налаштувань

- колекція розкладів містить усі дії які повинні бути виконані Arduino.

```
motherBoardId: { type: String, required: true },
deviceId: { type: String, required: true },
turnOn:{ type: Boolean, required: true },
type: { type: String, required: true },
time: { type: Date, required: true},
```

Рисунок 2.19 – Схема колекції розкладів

Для ідентифікації потрібної Arduino використовується motherBoardId, він відповідає за унікальний номер Arduino плати, адміністратор вводить його при монтування пристрою.

2.4 Мікросервісна архітектура

Мікросервісна архітектура та монолітна архітектура представляють два різних підходи до розробки програмного забезпечення. Давайте порівняємо їх за декількома ключовими аспектами та розглянемо основні ризики для обох підходів.

Монолітна архітектура це традиційна модель програмного забезпечення, яка побудована як уніфікована одиниця, самодостатня та незалежна від інших програм. Монолітна архітектура це окрема велика обчислювальна мережа з єдиною кодовою базою, яка об'єднує всі бізнес-завдання. Щоб внести зміни в програму такого типу, потрібно оновити увесь додаток, отримавши доступ до бази коду та створивши та розгорнувши оновлену версію інтерфейсу служби. Це робить оновлення складнішим та забирає багато часу[5].

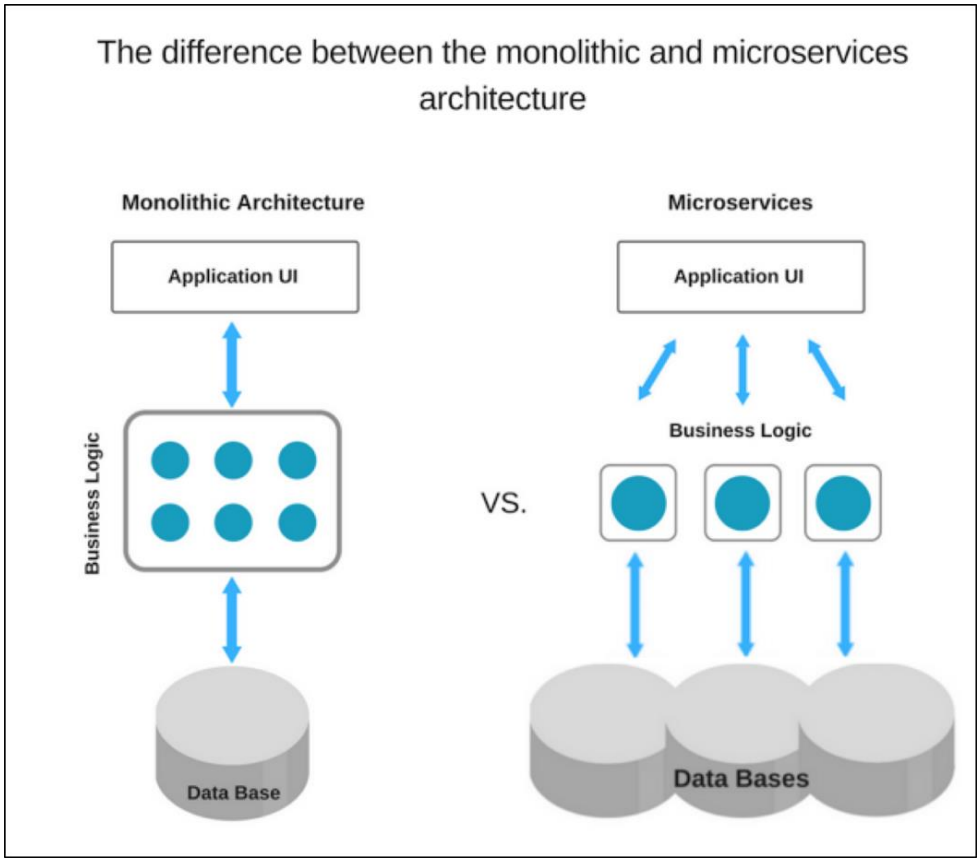

Рисунок 2.20 – Порівняння монолітної та мікросервісної архітектури

Мікросервісна архітектура це стиль архітектури програмного забезпечення, в якому програмний додаток складається з невеликих, незалежних і взаємодіючих служб, які називаються мікросервісами(рис.2.20). Кожен мікросервіс виконує конкретну функцію та має власну базу коду та базу даних. Вони взаємодіють між собою через відкриті API.

До переваг мікросервісної архітектури можна віднести:

- незалежність служб: кожен мікросервіс може бути розроблений, вдосконалений та масштабований незалежно від інших служб. Це дозволяє командам розробників працювати над конкретними функціями чи послугами без впливу на інші частини системи;
- спрощена розгортка і масштабування: мікросервіси можуть бути розгорнуті і масштабовані окремо, що полегшує управління інфраструктурою та забезпечує ефективне використання ресурсів;
- гнучкість технологій: кожен мікросервіс може бути розроблений із використанням різних технологій та мов програмування, залежно від його конкретної функціональності. Це дозволяє використовувати найкращі інструменти для конкретної задачі;
- покращена масштабованість: мікросервісна архітектура дозволяє ефективно масштабувати тільки ті служби, які потребують більше ресурсів, що забезпечує ефективне використання інфраструктури;
- легше управління кодовою базою: розділення системи на невеликі мікросервіси полегшує управління кодовою базою. Кожен мікросервіс може бути розроблений, тестований та підтримуваний окремо;
- зменшення впливу помилок: якщо один мікросервіс виявляється нестійким чи виникають проблеми, це не впливає на решту системи. Це полегшує виявлення, ізоляцію та виправлення помилок;

- легка інтеграція: мікросервіси спрощують інтеграцію нових функцій та сервісів. Вони можуть бути підключені через API, що полегшує спільну роботу інших сервісів.

Apache Kafka є розподіленою системою потокової обробки подій та повідомлень, яка широко використовується для будівництва масштабованих та надійних систем обміну даними в реальному часі. У мікросервісній архітектурі Kafka може використовуватися для ефективного обміну даними між різними мікросервісами[6]. Ось деякі ключові аспекти Kafka у мікросервісному середовищі:

- публікація-підписка (Publish-Subscribe): Kafka використовує модель публікації-підписки, де виробник (публікатор) публікує повідомлення в тему, і один чи декілька споживачів (підписники) можуть відстежувати цю тему та обробляти повідомлення;
- інтеграція мікросервісів: Kafka дозволяє різним мікросервісам взаємодіяти та обмінюватися даними. Кожен мікросервіс може бути виробником (надсилати дані) або споживачем (отримувати та обробляти дані) у Kafka-темах;
- зберігання подій: Сервіс може зберігати дані (події) протягом певного періоду часу або досягнення певного розміру. Це дозволяє споживачам обробляти події, які виникли під час їх відсутності або які важливі для аналізу;
- масштабованість: Kafka спроектована для масштабованості та високої доступності. Вона може працювати в розподіленому середовищі, що дозволяє легко масштабувати обробку даних залежно від потреб системи;
- надійність та відновлення стану: Kafka гарантує надійну доставку повідомлень, використовуючи реплікацію даних та механізми відновлення стану. Це важливо для забезпечення надійності обміну даними в системі мікросервісів;

• взаємодія з різними технологіями: використовувати сервіс можна незалежно від конкретної технології програмування чи мови. Це дозволяє різним мікросервісам, написаним на різних мовах, взаємодіяти через сервіс;

• легка інтеграція з іншими системами: сервіс може взаємодіяти з іншими системами та інфраструктурою, такою як бази даних, аналітичні системи та інші, що полегшує обмін даними та інтеграцію між різними компонентами системи[6].

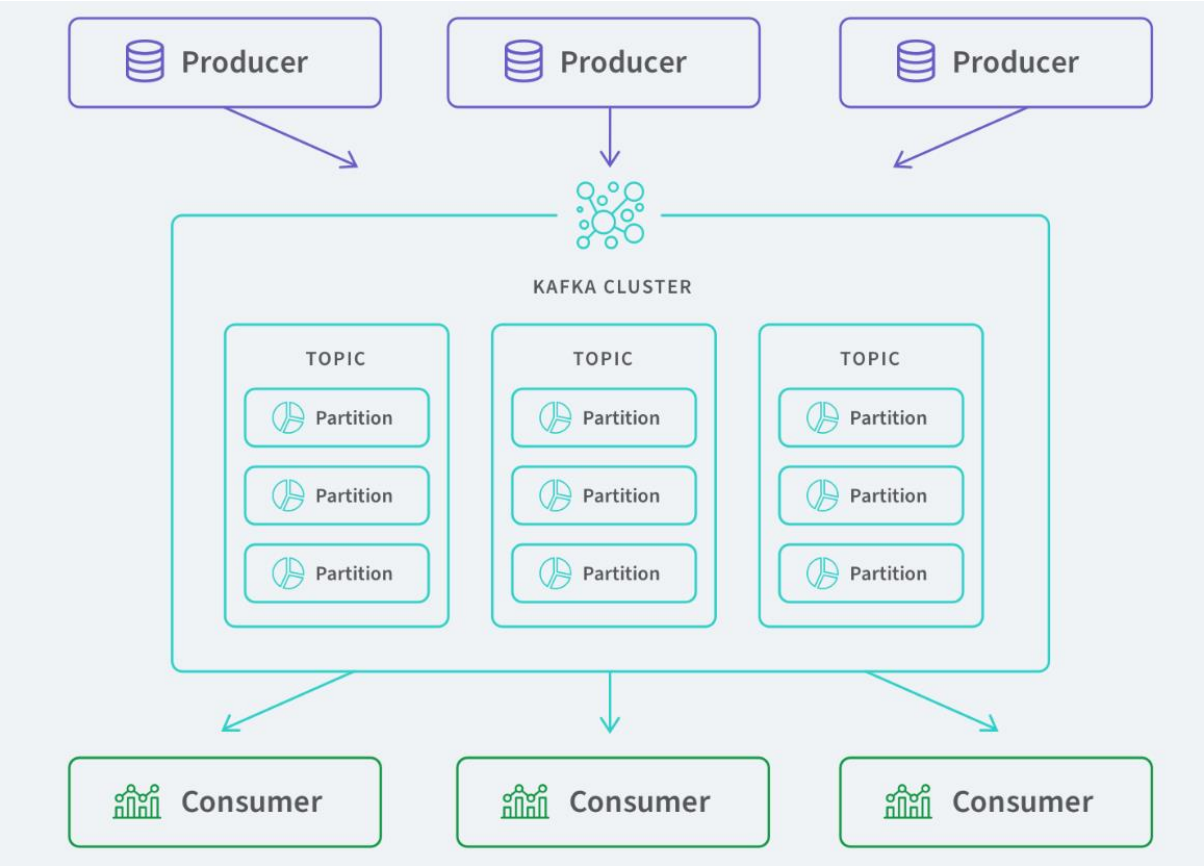

Рисунок 2.21 – Модель Apacahe Kafka

Використання Apache Kafka у мікросервісній архітектурі допомагає забезпечити ефективний та масштабований обмін даними між різними складовими системи, забезпечуючи високу доступність та надійність в реальному часі.

2.5 Структура мікросервісів

Для реалізації функцій бізнес логіки та зв’язку з UI і Arduino було розроблено 10 мікросервісів:

- мікросервіс API GATEWAY – виконує роль пропускного сервісуміж SPA та мікросервісів. Виконує перевірку запитів на повноту(наявність точені авторизації, відповідність заголовків), безпеку, доступність від front-end додатку до відповідного мікросервіса;
- мікросервіс Авторизації – відповідає за реєстрацію та авторизацію у системі, перевірку статусу та прав користувача;
- мікросервіс Замовлень – відповідає за створення замовлень на додавання, видалення, технічний огляд пристрою, зв'язок між користувачем та адміністраторами;
- мікросервіс Статистики – обробляє дані налаштувань пристроїв беручи до уваги геолокацію користувачів, завдяки цьому створює автоматичні налаштування на прикладі інших користувачів тільки поблизу вас;
- мікросервіс Погоди – отримує дані про поточні та майбутні погодні умови з сервісу OpenWeather API;
- мікросервіс Налаштувань – дозволяє користувачам налаштовувати пристрої як власноруч, так і за допомогою мікросервісу Статистики і мікросервісу Погоди;
- мікросервіс Електронної пошти – виконує надсилання електронних листів користувачам;
- мікросервіс Розкладу – перетворює налаштування які ав користувач чи система в зрозумілі виклики для Arduino та надсилає їх на мікросервіс Керування пристроями;

- мікросервіс Керування пристроями – служить MQTT брокером, дані публікує мікросервіс Розкладу, а кожен Arduino пристрій підписаний на відповідний потік даних;
- мікросервіс Обробки виключень – виконує функцію логера, оброблює помилки інших мікросервісів та записує у відповідні файли помилки та середовище в якому вони виникли.

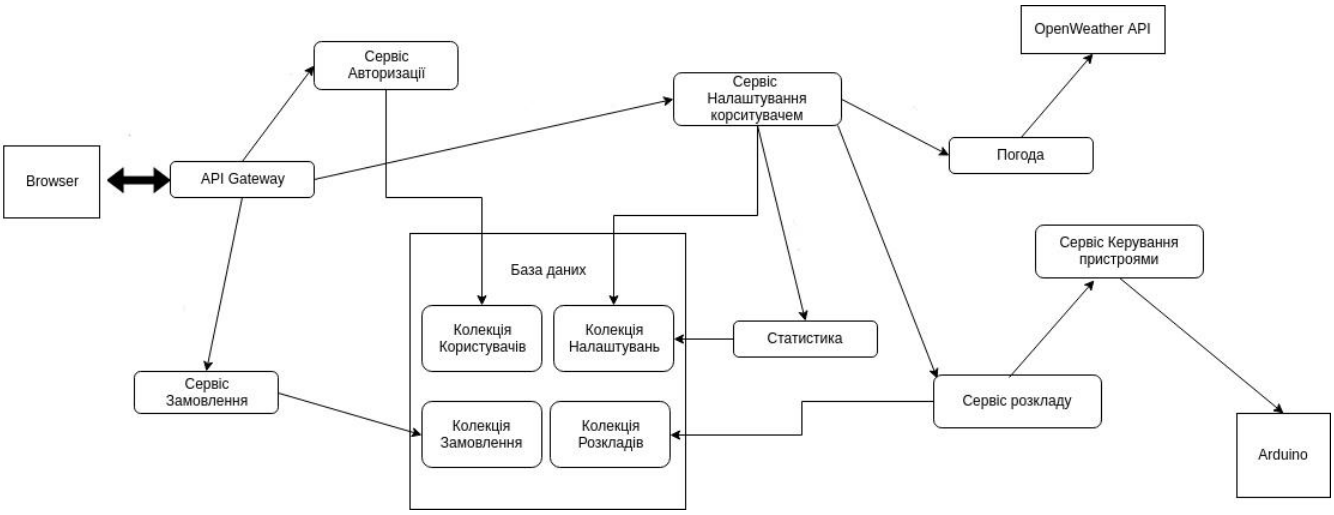

Рисунок 2.22 – Діаграма мікросервісів

Між собою мікросервіси спілкуються за допомогою HTTP протоколу. А мікросервіс Керування пристроями використовує MQTT для спілкування з Arduino, оскільки він використовує набагато менше ресурсів в порівнянні з HTTP, але у свою чергу має менше можливостей, але для взаємодії з Arduino їх цілком достатньо[5].

Використання Apache Kafka у мікросервісній архітектурі допомагає забезпечити ефективний та масштабований обмін даними між різними складовими системи, забезпечуючи високу доступність та надійність в реальному часі.

III. ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ СИСТЕМИ

3.1 Використання компонентів з відомими вразливостями

Сучасні Javascript додатки як клієнтські так і серверні, як і усі додатки використовують велику кількість зовнішніх бібліотек та компонентів. Майже усі бібліотеки є open-source проектами, тобто їх створюють та підтримують ентузіасти, одиниці підтримують компанії, це призводить до того що вони часто є достатньо протестовані і можуть мати чисельні вразливості.

У Front-end додатках існує три варіанти використання зовнішніх бібілотек чи компонентів:

- через CDN(Content delivery network);
- завантажити код бібілотеки та підключати його як звичайний js файл;
- npm(Node Package Manager).

Оптимальним з точки зору виявлення компонентів з потенційними вразливостями є npm. Оскільки він дозволяє створювати репорти про потенційні вразливості користувачам бібліотек, і у випадку якщо вони будуть там буде реальна вразливість ця версія бібліотеки буде позначена як вразлива з посиланням на причину.

За цим не потрібно слідкувати на сторінках бібліотек, при кожному виклику npm install утиліта сканує також на наявність вразливостей в використовуваній версії.

```
~/Web/secure-vue-app  master !1  npm install
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.3 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.3: wanted
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules/watchpack-choki
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wante
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules/webpack-dev-ser
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wante

audited 1359 packages in 2.757s

76 packages are looking for funding
  run `npm fund` for details

found 196 vulnerabilities (2 low, 96 moderate, 80 high, 18 critical)
  run `npm audit fix` to fix them, or `npm audit` for details
```

Рисунок 3.1 – Після встановлення пакетів отримуємо повідомлення про їх вразливості.

Для отримання детальнішої інформації про вразливості потрібно запустити команду npm audit.

```
                    === npm audit security report ===

# Run  npm install --save-dev @vue/cli-service@5.0.8  to resolve 47 vulnerabilities
SEMVER WARNING: Recommended action is a potentially breaking change
```

Low	**Prototype Pollution in node-forge debug API.**
Package	node-forge
Dependency of	@vue/cli-service [dev]
Path	@vue/cli-service > webpack-dev-server > selfsigned > node-forge
More info	https://github.com/advisories/GHSA-5rrq-pxf6-6jx5

Low	**URL parsing in node-forge could lead to undesired behavior.**
Package	node-forge
Dependency of	@vue/cli-service [dev]
Path	@vue/cli-service > webpack-dev-server > selfsigned > node-forge
More info	https://github.com/advisories/GHSA-gf8q-jrpm-jvxq

Рисунок 3.2 – npm audit

Результат команди містить доповідь по кожній вразливості, рівень загрози, який пакет містить цю загрозу, дерево залежності(хто підключає цей пакет), та детальний опис саме цієї вразливості.

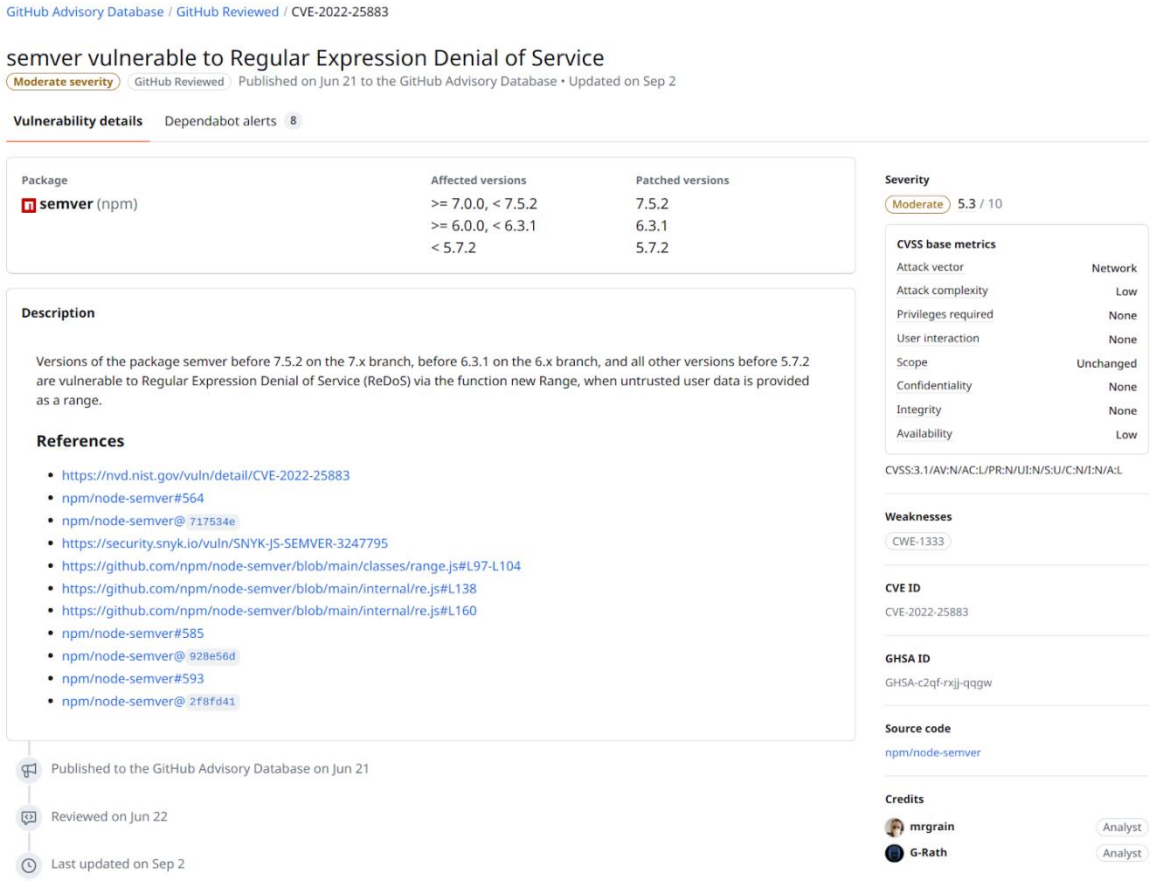

Рисунок 3.3 – Приклад опису вразливості.

Існує команда npm audit fix яка дозволяє автоматично оновити бібліотеки до безпечних версій, але це працює не завжди, якщо зміна версії major тоді це не буде оновлено автоматично, також після npm audit fix бібліотеки після оновлення можуть змінити свою поведінку.

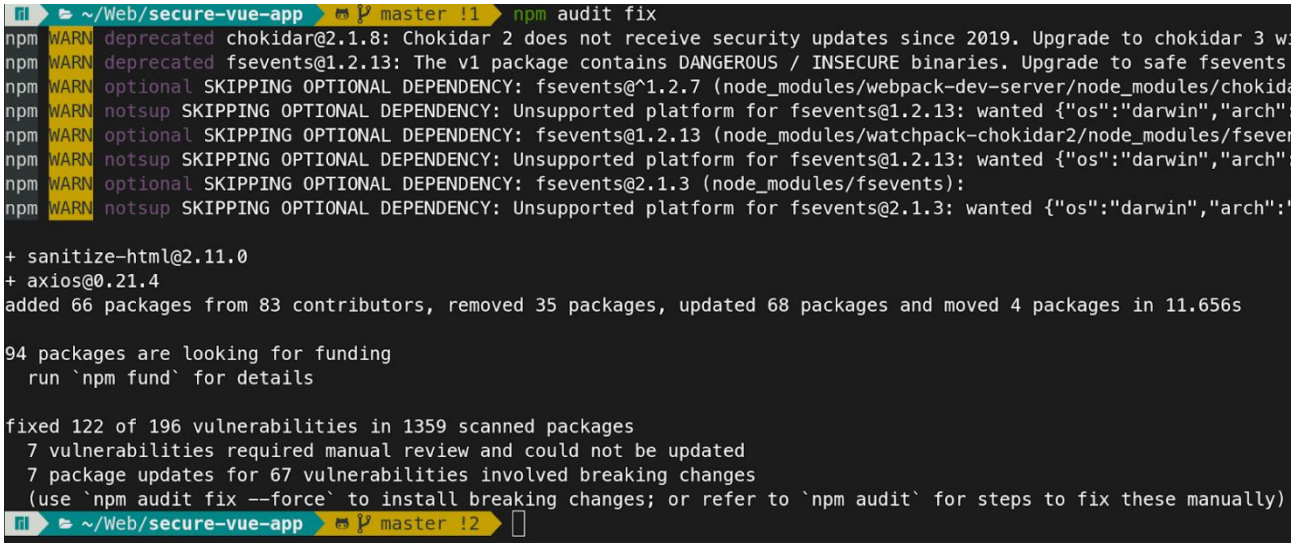

Рисунок 3.4 – npm audit fix

Проаналізовано ризики використання компонентів з відомими вразливостями, як перевіряти компоненти на наявність вразливостей, та як мігрувати до версій компонентів, які уже не вразливі.

3.2 Логування та відслідковування помилок

Sentry - це платформа для моніторингу та відлагодження програмного забезпечення, яка допомагає розробникам виявляти та вирішувати проблеми в їхньому коді, а також відстежувати помилки та проблеми продукції. Sentry допомагає командам розробників виявляти, реєструвати та аналізувати помилки, які виникають в їхніх додатках та веб-сервісах, і надає інформацію, необхідну для того, щоб швидко реагувати на ці проблеми.

Основні функції Sentry включають:

- збір та агрегація логів та винятків (exceptions) з додатків та сервісів.
- відстеження помилок та стеження за їхнім станом;
- надсилання сповіщень розробникам про виявлені проблеми;

- аналіз стеку викликів (stack traces) для ідентифікації місця виникнення помилок;
- інтеграція з іншими DevOps-інструментами, такими як системи контролю версій (Version Control Systems), інструменти неперервної інтеграції (Continuous Integration) та іншими.

Sentry допомагає розробникам підтримувати високу якість свого програмного забезпечення та забезпечує зручні інструменти для відлагодження та виправлення помилок в реальному часі, що допомагає забезпечити стабільну та надійну роботу програми.

Для використання sentry у клієнтському додатку потрібно додати пакет @sentry/browser і @sentry/integrations. Також потрібно отримати DSN ключ у sentry, створити аккаунт та оплатити підписку.

Важливо підключити sentry якомога швидше в додатку, щоб він міг зловити помилки під час збирання додатку. Для підключення потрібно викликати метод init та передати йому конфігурацію.

```
import Vue from 'vue';
import * as Sentry from '@sentry/browser';
import * as Integrations from '@sentry/integrations';

Sentry.init({
  dsn: 'your private key',
  integrations: [new Integrations.Vue({ Vue, attachProps: true, logErrors: true })],
});
```

Рисунок 3.5 – Встановлення sentry

Тепер усі помилки які не потрапили в catch будуть попадати на sentry сервер.

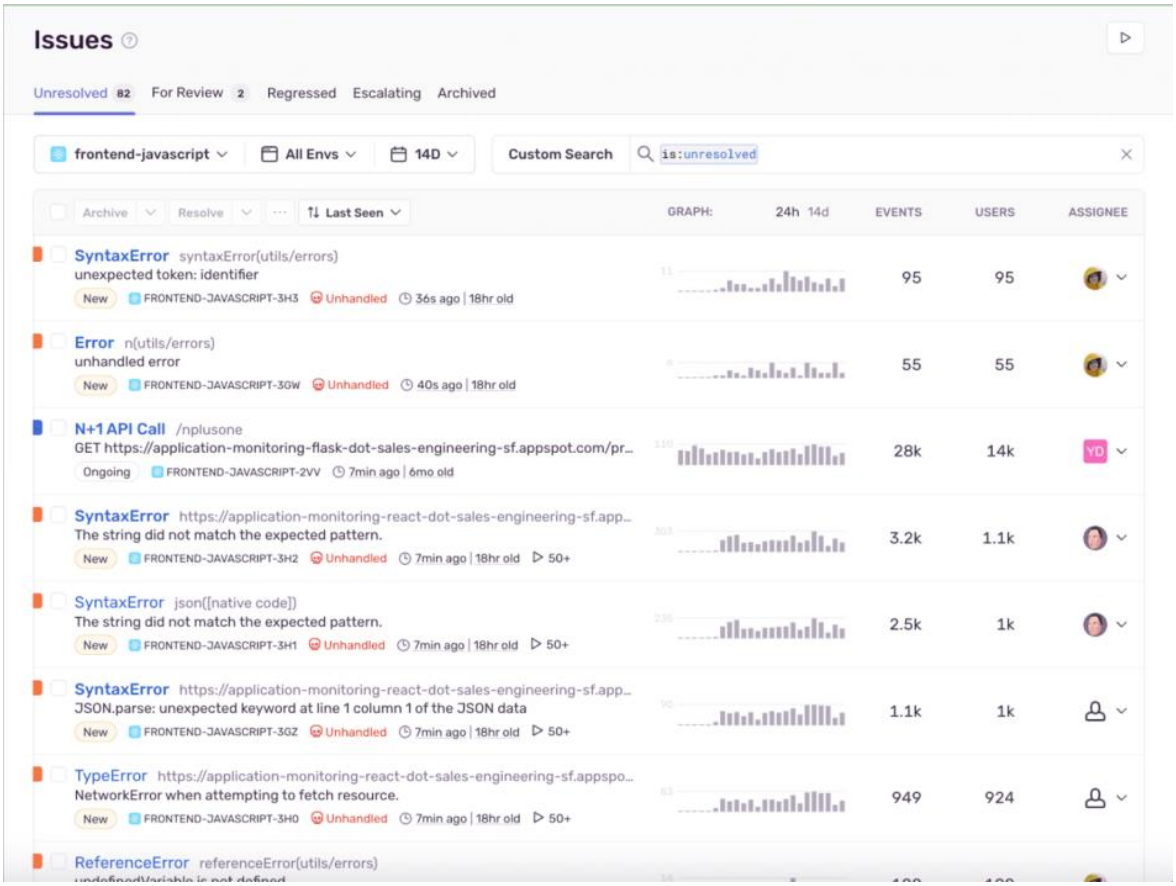

Рисунок 3.6 – Інтерфейс Sentry

Важливим аспектом у роботі додатку є логування та відслідковування помилок не тільки на серверній стороні, але і на клієнтській за допомогою Sentry.

3.3 Захист від DDOS атак

Важливим аспектом є захист від DDOS атак на додатку, для захисту клієнтської частини можна використати Cloudflare. Cloudflare відомий надійною системою захисту від розподілених атак на відмову в обслуговуванні (DDoS), він дозволяє попередити таку атаку, та захисти сайт.

Ключові аспекти захисту Cloudflare від DDoS:

Cloudflare відомий своєю надійною системою захисту від розподілених атак на відмову в обслуговуванні (DDoS), спрямованою на зменшення та

відсторонення різних видів таких атак. Ось деякі ключові аспекти захисту Cloudflare від DDoS:

- Cloudflare веде широку мережу дата-центрів по всьому світу, стратегічно розташованих в різних регіонах. Ця мережа Anycast дозволяє ефективно маршрутизувати трафік та уникати DDoS-атаки, розподіляючи навантаження між декількома серверами;
- аналізує вхідний трафік в реальному часі та використовує алгоритми для розрізнення між легітимними запитами користувачів та зловмисним трафіком, пов'язаним із DDoS-атакою. Ідентифікуючи та ігноруючи запити які виконують атаку(рис.3.7);

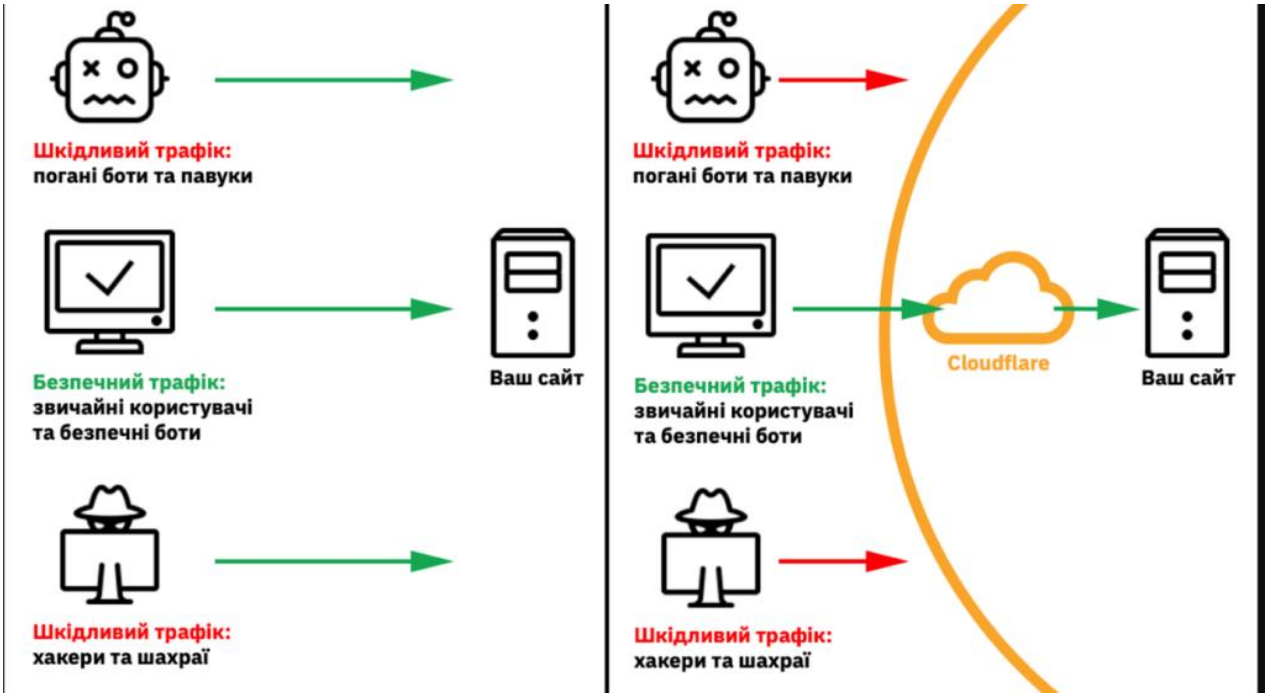

Рисунок 3.7 – Cloudflare перевіряє запити на можливість DDOS атаки

- дозволяє налаштовувати правила лімітування швидкості, які обмежують кількість запитів, які можуть надходити з певної IP-адреси за визначений період часу;

- використовує аналіз поведінки для виявлення аномалій у поведінці користувачів та паттернах трафіку. Розуміючи нормальну поведінку користувачів, сервіс може ідентифікувати та блокувати аномалії, які можуть свідчити про DDoS-атаку;
- у випадку коли виявляється підозрілий трафік, сервіс може застосовувати рішення для перевірки на людину, такі як відображення відвідувачам CAPTCHA. Реальні користувачі можуть легко пройти такі виклики, тоді як боти ні;
- захист охоплює і DNS, що допомагає запобігти атакам, спрямованим на DNS.

3.4 Ризики використання штучного інтелекту.

Існує ризик використання штучного інтелекту для створення прогнозованих налаштувань, ризики які виникають при використанні штучного інтелекту можуть проявитись не тільки у випадку некоректної поведінки програми через некоректні отримані дані, але також важливим є ризик порушення GDPR, яке може виникнути у випадку використання інформації користувачів без їх згоди, або використання інформації не за призначенням[9]. Це може понести серйозні збитки, не тільки судові справи, але і репутаційні ризики. Щоб уникнути ризиків з GDPR(General Data Protection Regulation) потрібно керуватись наступними принципами:

- обмеження використовуваної інформації користувачів, важливо використовувати тільки ті дані які потрібні для навчання моделі і виключно у випадку коли користувач надає згоду на їх опрацювання. Також важливо пам’ятати що потрібно отримати згоду користувача для використання його особистої інформації для використання під час навчання моделі;
- мінімізація даних і обмеження зберігання, цей принцип вимагає мінімізації кількості, деталізації та тривалості зберігання особистої інформації

у наборі даних для навчання. Не збирайте та не копіюйте непотрібні дані для свого набору даних, якщо це не є обов'язковим для навчання. Також важливо зробити дані анонімними, де це можливо, якщо повна анонімність неможлива,потрібно зменшити деталізацію даних у наборі даних;

3.5 Ризики JWT авторизації.

JWT (JSON Web Token) є популярним механізмом авторизації в розподілених системах. Однак використання JWT пов'язане із певними ризиками, які можуть виникнути при неправильному впровадженні або конфігурації.

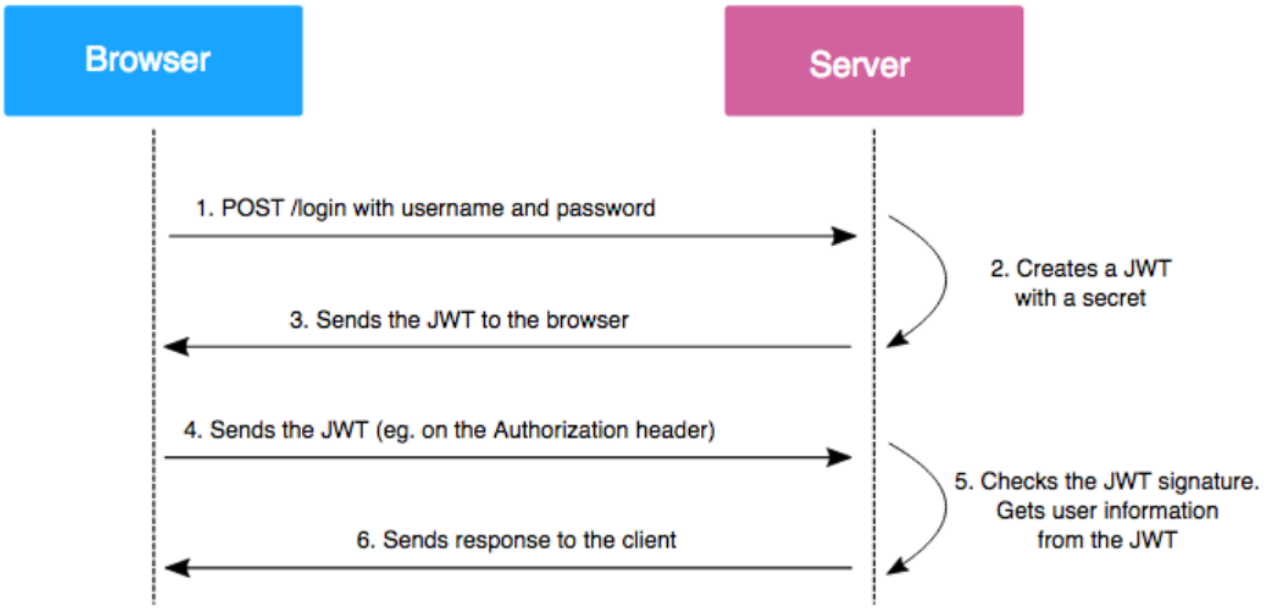

Рисунок 3.8 – Модель JWT авторизації

Використання JWT має наступні ризики та вразливості

- відкрите зберігання ключів: ссекретний ключ використовується для підпису JWT, і він повинен бути добре захищеним від несанкціонованого

доступу. Якщо зловмиснику вдасться отримати доступ до секретного ключа, це може призвести до компрометації всіх токенів, які підписані цим ключем. Зберігайте секретні ключі безпечно та оберігайте від несанкціонованого доступу, також використовуйте механізми управління секретами

- терміни дії токенів: тривалість життя токена – це період часу, протягом якого токен вважається дійсним. Короткі терміни дії токенів можуть допомогти зменшити ризики безпеки. Великий термін дії може вести до погіршення безпеки. Використовуйте оновлення токенів та механізми оновлення токену, наприклад використання Refresh Token;

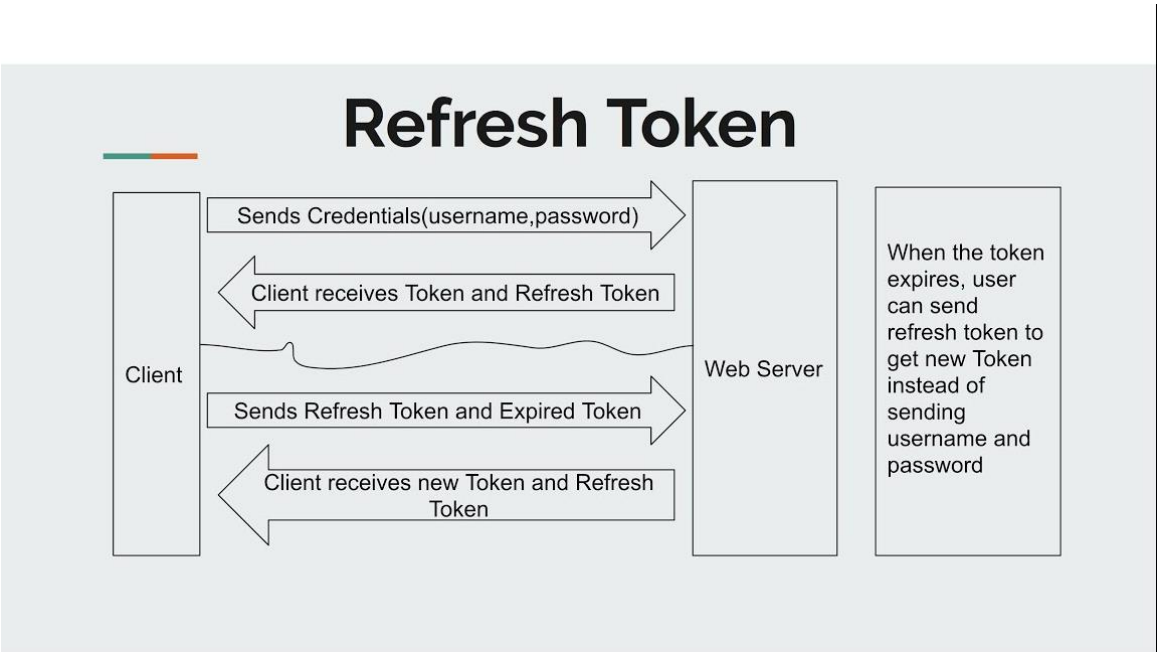

Рисунок 3.10 – Механізм Refresh Token

- відсутність шифрування: використовуйте HTTPS для захисту трафіку між клієнтом і сервером Також варто зашифрувати токени, якщо вони містять конфіденційну інформацію, за допомогою JWE (JSON Web Encryption);
- атака повторного використання токенів (Replay Attacks): це атаки, при яких зловмисник отримує попередній токен користувача і використовує його повторно для отримання доступу або виконання дій, на які він не має достатніх

прав. Варто використовувати timestamp або інші механізми для перевірки часу дії точена;

- витік інформації: обмежте інформацію, яка зберігається в JWT, до необхідного мінімуму, також уникайте збереження конфіденційної інформації в токенах;
- використання слабких алгоритмів підпису: уникайте використання застарілих або вразливих алгоритмів.JWT авторизації є достатньо захищеним рішенням, але все ж має певні вразливості, важливо уникати їх, оскільки наслідки від несанкціонованого доступу можуть бути критичними.

IV. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Метою кваліфікаційної роботи магістра є дослідження ризиків та вразливостей системи керування розумним будинком. Зважаючи на те, що проведення робіт з дослідження ризиків та вразливостей передбачає використання комп'ютерної техніки, а саме ПК та периферійних пристроїв, то є обов'язковим дотримання всіх вимог техніки безпеки і охорони праці.

Охорона праці при роботі з розумним будинком включає в себе ряд заходів та підходів для забезпечення безпеки та комфорту користувачів, а також для уникнення можливих ризиків та непорозумінь.

Оскільки розумний будинок використовує електронні пристроями важливо звертати увагу на електробезпеку. Потрібно впевнитись, що всі електричні пристрої та системи встановлені та підключені правильно. Уникайте перевантаження електричних мереж та використання неякісних електричних пристроїв.

Для того щоб пристрої працювали правильно і не створювали ризиків для життя та майна потрібно регулярно оновлювати програмне забезпечення пристроїв, також уникайте використання застарілих та вразливих систем

Уникайте розташування обладнання в місцях, де може відбутися його пошкодження. Захищайте пристрої від впливу вологи та температурних змін.

Для ефективної і безпечної роботи колективу працівників з розробки ПЗ комп'ютерних систем, в тому числі і фахівців з дослідження методів та інструментальних засобів, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці

Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за №

207 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями».

Згідно Вимог приміщення, де розміщені робочі місця операторів, крім приміщень, у яких розміщені робочі місця операторів великих ЕОМ загального призначення (сервер), мають бути оснащені системою автоматичної пожежної сигналізації відповідно до цих вимог: переліку однотипних за призначенням об'єктів, які підлягають обладнанню автоматичними установками пожежогасіння та пожежної сигналізації, затвердженого наказом Міністерства України з питань надзвичайних ситуацій та у справах захисту населення від наслідків Чорнобильської катастрофи від 22.08.2005 N 161, зареєстрованого в Міністерстві юстиції України 05.09.2005 за N 990/11270 (НАПБ Б.06.004-2005);

Державних будівельних норм "Інженерне обладнання будинків і споруд. Пожежна автоматика будинків і споруд", затверджених наказом Держбуду України від 28.10.98 N 247 (далі - ДБН В.2.5-56:2014, з димовими пожежними сповіщувачами та переносними вуглекислотними вогнегасниками[14].

В інших приміщеннях допускається встановлювати теплові пожежні сповіщувачі. Приміщення, де розміщені робочі місця операторів, мають бути оснащені вогнегасниками, кількість яких визначається згідно з вимогами ДСТУ 4297:2004 «Пожежна техніка. Технічне обслуговування вогнегасників». Загальні технічні вимоги і з урахуванням граничнодопустимих концентрацій вогнегасної рідини відповідно до вимог НАПБ А.01.001-2014.

Організація робочого місця фахівця із дослідження повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Лінія електромережі для живлення комп’ютера та периферійних пристроїв повинні бути виконаними як окрема групова трьох провідна мережа

шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток електроживлення. Не допускається підключати на щиті до одного контактного затискача нульовий робочий та нульовий захисний провідники.

Комп’ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. У штепсельних з'єднаннях та електророзетках, крім контактів фазового та нульового робочого провідників, мають бути спеціальні контакти для підключення нульового захисного провідника. Порядок роз’єднання при відключенні має бути зворотним.

Не допускається підключати комп’ютери до звичайної двох провідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення комп’ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В.

Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

У результаті аналізу вимог щодо охорони праці користувачів комп’ютерів, визначено особливості організації робочих місць, вимог з природного та штучного освітлення, електробезпеки, для ефективної і безпечної роботи фахівців.

4.2 Комп’ютерне забезпечення процесу оцінки радіаційної та хімічної обстановки

Екологічне співтовариство розробило сімейство інструментів комплексної екологічної оцінки. Програмне забезпечення і послуги (ESS), комерційна група IIASA, включаючи AirWare (для повітряних проблеми якості), WaterWare (для якості води), CityWare (якість повітря і води в контексті великих міст) і EIAxpert (для надання допомоги із загальним впливом на навколишнє середовище). Функціональність в цілому схожа на RAISON, хоча з великим акцентом на моделювання і меншим акцентом на керування даними. Знову ж таки, інструменти ESS розроблені як модульні набори інструментів (доступні спеціальні системи для вирішення конкретних завдань). Компоненти включають стандартні імітаційні моделі, включаючи моделі ISC і PBM Агентства з охорони навколишнього середовища США, управління даними, в тому числі ГІС, аналіз даних (наприклад, аналіз часових рядів даних спостережень), візуалізація, а також оптимізація [15].

Іноді немає готових моделей, придатних для конкретного застосування, але тягар розробки нової програми на Фортрані або C / C ++ є надмірним. Розробка моделі оточення може відносно легко реалізувати власні моделі комп'ютерів і не турбуватися про включення процедур для вирішення рівнянь, візуалізації і т. д. Як правило, за допомогою цих інструментів користувач просто повинен вказати свою модель, використовуючи або математичні рівняння, або спеціальні графічні символи або значки, які безпосередньо представляють поведінку системи.

На даний момент є розроблені моделі комп’ютерного забезпечення процесу для оцінки радіаційної та хімічної обстановки.

GEMS – це система на основі моделей, яка підтримує оцінки схильності і ризику, надаючи доступ до одиночних і мультимедійних моделям експозиції, фізико-хімічні властивості методи оцінки, статистичний аналіз, графічні та

картографічні програми з відповідними даними на навколишнє середовище, джерела, рецептори і популяції. У розробці з 1981 року, GEMS надає аналітикам інтерактивний, легко досліджуваний інтерфейс для різних моделей, програм і даних, які необхідні для оцінки хімічного впливу і ризику [14].

HSPF – це комплексний пакет для моделювання кількості і якості стоків з багатоцільових водозборів і процесів радіації, що відбуваються в потоках або повністю змішаних озерах. Це дозволяє інтегроване моделювання землі і ґрунту, процесів забруднення при гідравлічній і осадово-хімічній взаємодії. Результатом моделювання є тимчасові дані витрати стоку, концентрація поживних речовин і пестицидів, а також дані кількості і якості води в будь-якій точці водозбору. Алгоритми якості води включають динаміку BOD / DO, вуглець, азот і фосфор. Процеси трансформації, які включені в модель це: гідроліз, фотоліз, окислення, випаровування, сорбція і біодеградація. Вторинні або «дочірні» хімічні речовини також моделюються.

Вимоги до даних для моделі можуть бути досить широкими в залежності від конкретного застосування.

Модель MMSOILS – це методологія оцінки впливу на людину і ризику для здоров'я, пов'язаних з викидами забруднень з небезпечних відходів. Мультимедійна модель, що стосується перенесення хімічної речовини в ґрунтові води, поверхневі води, атмосферу і накопичення в їжі. Шляхи впливу на людину, які розглянуті в методології включають: потрапляння в грунт, вдихання летких речовин в повітря і тверді частинки, шкірний контакт, прийом питної води і т.д. Ризик, пов'язаний із загальною дозою опромінення, розраховується на основі хімічної токсичності [15].

ВИСНОВКИ

Під час виконання кваліфікаційної роботи проведено аналіз актуальності дослідження ризиків та вразливостей системи керування розумного будинку та досліджено існуючі дослідження.

Існуючі рішення приділяють недостатньо уваги вразливостями в архітектурний рішень та їх програмній реалізації, значним ризиком є Memory Leak при використанні SSR архітектури на стороні клієнта, при використання SPA ці ризики зменшуються.

Предметом дослідження було обрано дослідження ризиків та вразливостей систем керування розумним будинком.

У другому розділі розглянуто потенційні архітектурні рішення для проектування системи керування розумним будинком. Розглянуто ризики які випливають з вибору тої чи іншої архітектури, переваги та недоліки кожної для системи керування розумним будинком. Також спроектовано структуру мікросервісів та колекції MongoDB, обрано рішення які будуть використані для проектування.

В третьому розділі розглянуто можливі ризики в системі керування розумним будинком, які наслідки вони можуть мати для продукту, також запропоновано рішення, для уникнення або зменшення наслідків від ризиків.

Система керування розумним будинком є складною системою з великою кількістю ризиків, оскільки вона використовує клієнт-серверну архітектуру, котра має значні ризики як на клієнтській так і на серверній частині продукту. Також варто окремо згадати про ризики пов’язані з використання штучним інтелектом, як програмні так і юридичні.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. David Flanagan JavaScript: The Definitive Guide: Master the World's Most-Used Programming Language 7th Edition, UCSB 2020.704.

2. Kyle SimpsonYou Don't Know JS: Async & Performance 2015, O'Reilly 136-154c.

3. Thomas Hunter II Distributed Systems with Node.js: Building Enterprise-Ready Backend Services 2020. 98-130c.

4. Shannon Bradshaw MongoDB: The Definitive Guide: Powerful and Scalable Data Storagme, O'Reilly 2019. 43-76c.

5. Sam Newman Building Microservices: Designing Fine-Grained Systems, O'Reilly 2022 21-45c.

6. Neha Narkhede, Gwen Shapira, Todd Palino Kafka: The Definitive Guide, O'Reilly 2017. 123–160c.

7. Robert C. Martin: Clean Code: A Handbook of Agile Software Craftsmanship, IEEE 2009. 464c.

8. Sarah Horton Learning Algorithms: A Programmer's Guide to Writing Better Code. 1st Ed, 2014. 280c.

9. Goller C., Küchler A. Learning task-dependent distributed representations by backpropagation through structure. Neural Networks, 1996., IEEE 374-394c;

10. OWASP. URL: https://owasp.org/ (дата звертання: 07.12.2023).

11. AJAX Systems: https://ajax.systems/ua/ (дата звертання: 07.12.2023).

12. Nuxt Docs. URL: https://nuxt.com/ (дата звертання: 07.12.2023).

13. Can I use. URL: https://caniuse.com/ (дата звертання: 07.12.2023).

14. Зеркалов Д.В. Охорона праці в галузі: Загальні вимоги. Навчальний посібник. К.: Основа. 2011. 551 с.

15. Толок А.О. Крюковська О.А. Безпека життєдіяльності: Навч. посібник. – 2011. – 215 с.

ДОДАТОК А

Тези конференції

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Університет імені П'єра і Марії Кюрі (Франція)
Маріборський університет (Словенія)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Міжнародний університет цивільної авіації (Марокко)
Наукове товариство ім. Т.Шевченка

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник
тез доповідей

XII Міжнародної науково-практичної конференції молодих учених та студентів
6-7 грудня 2023 року

УКРАЇНА
ТЕРНОПІЛЬ – 2023

Матеріали XII Міжнародної науково-практичної конференції молодих учених та студентів «АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ» – Тернопіль, 6-7 грудня 2023 року

УДК 004.031.6

О. Р. Оробчук, доктор філософії, І. М. Кивацький
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

АНАЛІЗ РИЗИКІВ ТА ВРАЗЛИВОСТЕЙ В СИСТЕМАХ КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ

O. R. Orobchuk, Dr, I. M. Kyvatskyi

ANALYSIS OF RISKS AND VULNERABILITIES IN SMART HOME MANAGEMENT SYSTEMS

На сьогодні кількість розумних будинків у світі налічує більше 500 мільйонів, що робить повсякденне життя набагато зручнішим. Однією з найбільших переваг технологій розумного будинку є використання пристроїв, підключених до Інтернету, для дистанційного захисту особистих помешкань. Незважаючи на те, що розумні домашні пристрої безпеки забезпечують легкість захисту будинків від крадіжки, пошкодження чи нещасного випадку, вони також створюють ризик зниження безпеки персональних даних.

Розумні будинки вразливі до різного роду кібератак через вразливі локальні мережі та недосконалі пристрої IoT. В цьому контексті безпеку розумного пристрою розглядатимемо як функцію управління ризиком того, що пристрій буде зламано, з урахуванням збитків внаслідок зламу, а також часу і ресурсів для забезпечення необхідного рівня захисту.

Загалом, можна виокремити такі ризики, як ботнети, man-in-the-middle атаки, DoS-атаки, крадіжки даних, дистанційний запис, централізація мереж IoT. Перехоплення та злом шифрування є найпоширенішими способами хакерського проникнення в мережу. Під час атаки хакери викрадають будь-який пакет даних, що передається між пристроєм і маршрутизатором, передають його на свій пристрій і використовують brute-force атаку, щоб розшифрувати його. Зазвичай це займає лише хвилини. Wi-Fi може бути вразливим до атак через стандартні або слабкі SSID або паролі та вразливі протоколи шифрування. Облікові дані за замовчуванням дозволяють зловмиснику отримати доступ до маршрутизатора без зусиль. Надійні паролі Wi-Fi змушують хакерів шукати складніші шлюзи для проникнення в мережу. Більшість маршрутизаторів Wi-Fi використовують WEP (Wired Equivalent Privacy), WPA (Wi-Fi Protected Access) або протокол безпеки WPA2. WEP — це потоковий шифр RC4. Слабкою стороною WEP є малий розмір вектора ініціалізації (24-розрядний IV), що спричиняє його повторне використання. Це повторення робить його вразливим.

Постачальники IoT не можуть надати необхідні спеціальні рішення безпеки. Крім того, розумні домашні пристрої часто працюють під управлінням невеликих операційних систем, таких як INTEGRITY, Contiki, FreeRTOS і VxWorks, чиї рішення безпеки не такі надійні, як рішення систем на базі Windows або Linux. Оскільки розумні домашні пристрої все більше розширюються у функціональності, розуміння ризиків безпеки персональних даних і способів їх зменшення є критично важливим і вимагає поєднання технологій, процесів і політик. Важливо, щоб пристрої мали вбудовану безпеку як основну, а не як додаткову функцію.

Література

1. Джермен Галегуа. Розумні міста — ArtHuss 2021. — 129 p.
2. Mads Kristensen. The Automated Home: A practical guide to automating your smart home – London : A Book Apart 2023 – 55 p.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.031.6

О.Р. Оробчук, доктор філософії, І.М. Кивацький
Тернопільський національний технічний університет імені Івана Пулюя

ДОСЛІДЖЕННЯ СТРАТЕГІЙ КІБЕРЗАХИСТУ СИСТЕМ КЕРУВАННЯ РОЗУМНИМ БУДИНКОМ

O.R. Orobchuk, Dr, I.M. Kyvatskyi

RESEARCH OF CYBER PROTECTION STRATEGIES OF SMART HOME CONTROL SYSTEMS

Технології розумного будинку з року в рік демонструють значний прогрес. Але разом з цим все більш актуальним стають і проблеми кібербезпеки. Стратегії кіберзахисту систем керування розумним будинком включають комплекс заходів для запобігання, виявлення та відповіді на можливі кіберзагрози які можуть виникати під час користування розумним будинком.

Важливим аспектом кіберзахисту розумного будинку є авторизація та аутентифікація, у OWASP Top 10 2021 згадано про Broken Access Control і Identification and Authentication Failures як одні з найчастіших вразливостей, як за кількістю так і за можливими збитками.

Можливою стратегією для забезпечення більшої надійності є використання двофакторної аутентифікації – це суттєво може ускладнити несанкціонований доступ.

Згідно OWASP TOP 10 2021 Cryptographic Failures посідає друге місце серед вразливостей, це означає, що для передачі даних між пристроями та сервером потрібно використовувати виключно шифрований зв'язок згідно стандартам, уникаючи використання застарілих методів шифрування та протоколів зв'язку.

Важливо використовувати в системі актуальні версії зовнішніх бібліотек та фреймворків, уникаючи використання компонентів з відомими вразливостями, для цього потрібно використовувати спеціальне програмне забезпечення яке дозволяє відслідковувати появу вразливостей в версіях компонентів та шляхи оновлення до безпечних версій. Більш ефективним буде включення автооновлення прошивки.

Для відслідковування збоїв та можливих атак варто використовувати моніторинг та логування, яке дозволяє відслідковувати дії в системі, та у випадку виникнення помилки чи атаки отримати кроки для її відтворення та подальшого вирішення.

Відстежувати цифрові підписи вхідного трафіку та попереджати про виявлення шкідливого контенту дозволить інвестування в надійний брандмауер та використання VPN.

Моніторинг дозволяє переглядати навантаження на систему, і при потребі підключати нові інстанси. При правильно налаштованому моніторингу можна швидко дізнатись про DDOS атаку на сервер.

Сучасною стратегією є також використання технології блокчейн, яка може забезпечити безпечну та децентралізовану платформу для пристроїв IoT.

Для організацій актуальною буде розробка і застосування політики безпеки з описом оптимальних методів захисту пристроїв IoT, а також її регулярний перегляд і оновлення.

Література

1. Джермен Галегуа. Розумні міста — ArtHuss 2021. — 129 р.
2. Кеньо Г. В., Хома В. В.. Моделювання розумного будинку в середовищі Cisco Packet Tracer – Львів : Львівська політехніка 2022 – 104 р.